THE UNIVERSITY
of EDINBURGH

A formal language for statistical inference of uncertain stochastic systems

Anastasios-Andreas Georgoulas

Doctor of Philosophy

Laboratory for Foundations of Computer Science

School of Informatics

University of Edinburgh

2016

Abstract

Stochastic models, in particular Continuous Time Markov Chains, are a commonly employed mathematical abstraction for describing natural or engineered dynamical systems. While the theory behind them is well-studied, their specification can be problematic in a number of ways. Firstly, the size and complexity of the model can make its description difficult without using a high-level language. Secondly, knowledge of the system is usually incomplete, leaving one or more parameters with unknown values, thus impeding further analysis. Sophisticated machine learning algorithms have been proposed for the statistically rigorous estimation and handling of this uncertainty; however, their applicability is often limited to systems with finite state-space, and there has not been any consideration for their use on high-level descriptions. Similarly, high-level formal languages have been long used for describing and reasoning about stochastic systems, but require a full specification; efforts to estimate parameters for such formal models have been limited to simple inference algorithms.

This thesis explores how these two approaches can be brought together, drawing ideas from the probabilistic programming paradigm. We introduce ProPPA, a process algebra for the specification of stochastic systems with uncertain parameters. The language is equipped with a semantics, allowing a formal interpretation of models written in it. This is the first time that uncertainty has been incorporated into the syntax and semantics of a formal language, and we describe a new mathematical object capable of capturing this information. We provide a series of algorithms for inference which can be automatically applied to ProPPA models without the need to write extra code. As part of these, we develop a novel inference scheme for infinite-state systems, based on random truncations of the state-space. The expressive power and inference capabilities of the framework are demonstrated in a series of small examples as well as a larger-scale case study. We also present a review of the state-of-the-art in both machine learning and formal modelling with respect to stochastic systems. We close with a discussion of potential extensions of this work, and thoughts about different ways in which the fields of statistical machine learning and formal modelling can be further integrated.

Lay Summary

Writing a mathematical description of a natural or engineered system is common practice in the physical sciences. These *models* express our understanding of how the system works and are useful even if they are not necessarily fully detailed. For instance, they can be used to simulate the system and predict how it will behave under different circumstances. When the process being modelled is complicated, some researchers prefer to use specialised languages which let them describe the workings of the model more concisely. Before a model can be used, however, it must be tuned so that its behaviour matches the one observed in the real system. This can be a difficult and lengthy process. Statistical tools and methods exist for properly fitting models to observed data, but they are not easy to use by non-experts.

This thesis focuses on a specific kind of model which is widely used to describe systems that are not deterministic, that is, that behave differently each time because they make random decisions. We develop a language which can describe such systems even when some parts of them are not known. We also develop methods for fitting these systems to data, beyond what could be done with current tools. The result is a software tool that can be used to write models which include uncertain parts, and then automatically analyse observed data in order to refine this uncertainty. We believe that this can be useful for experts in fields like biology, ecology or business, for describing processes they encounter and fitting them to their data in an automated way.

Acknowledgements

Tradition or not, I must start by giving thanks to both my supervisors, Jane Hillston and Guido Sanguinetti. Their advice, experience and support has been invaluable and I am very grateful for their constant involvement in my project. Had it not been for them guiding, asking, pushing and steering me, I am not sure how I would have fared against what on more than one occasion appeared to be an insurmountable task.

I have been fortunate to be supported by Microsoft Research Cambridge throughout most of my PhD, and for that I thank them. During this time I have also had the pleasure to interact with many experts whom I wish to thank. Luca Cardelli, Andy Gordon and Charles Sutton have offered advice on various panels and reviews. Conversations with Luca Bortolussi, Mark Girolami, Maurizio Filippone and Chris Sherlock have proved very useful and fruitful. I owe thanks to Allan Clark and Vinayak Rao for sharing or publishing code which I later used or experimented with. Carron Shankland and Dalila Hamami were very helpful in identifying, preparing for and exploring the case study. I also wish to thank my examiners, Iain Murray and Verena Wolf, for their very useful insight during the viva, their comments and suggestions.

I don't think I could have wished for a better working environment than what I enjoyed in Informatics, and the members of the PEPA group and the Sanguinetti lab played a huge part in that, on both the professional and personal level; I apologise for not listing everyone individually, but it both saves space and I would inevitably leave someone out. Equally responsible for me making it to this point are all my officemates, and there have been many: Dimitris, Chris, Alireza, Cheng, Ludovica, Maria, Natalia, Moussa, Philippa and (left for last because she outranks everyone, of course) Yota — and anyone I have accidentally forgotten but who made 3.50 a happy, crazy place to work, and turned from colleague to friend.

I am always grateful for the support of my old friends, whether they are at home or abroad, but Vasia deserves a particular shout-out, if only for the mutual whining therapy sessions. I could most definitely not have done this without my close friends here: I cannot give enough thanks to Alex, İme, Jasmine, Ioanna, Giorgos and to Pete. You all kept me grounded, safe and sound, a true family. I could write a paragraph for each of you (but you're getting lumped together. Sorry.)

And, finally, I cannot say how much I owe to my parents Meropi and Kostas, and my sister Dido. No words will be enough, so I will simply say that I don't think you realise how much you have done for me.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified. The material presented in Chapter 3 has previously appeared in [40]. The material presented in Chapter 4 has previously been published in [42].

(*Anastasios-Andreas Georgoulas*)

In memory of Yanno Patrinos (1939-2015).

List of Abbreviations

ABC	Approximate Bayesian Computation
CMC	Constraint Markov Chain
CME	Chemical Master Equation
CTMC	Continuous Time Markov Chain
DTMC	Discrete Time Markov Chain
ESS	Effective Sample Size
FFBS	Forward Filtering - Backward Sampling
IMC	Interval Markov Chain
LNA	Linear Noise Approximation
LTS	Labelled Transition System
M-H	Metropolis-Hastings
MCMC	Markov Chain Monte Carlo
MJP	Markov Jump Process
ODE	Ordinary Differential Equation
pCMC	Probabilistic Constraint Markov Chain
pCTMC	Population Continuous Time Markov Chain
pdf	Probability Density Function
pMJP	Population Markov Jump Process
PSRF	Potential Scale Reduction Factor
SDE	Stochastic Differential Equation
SPA	Stochastic Process Algebra
SSA	Stochastic Simulation Algorithm

Table of Contents

Chapter 1

Introduction

Computational modelling is an established way to study and analyse systems, both those occurring in the natural world and engineered ones. The intention is to *abstract* away from the full system, keeping only those details that are well-known or important for the goal. A mathematical description of a complex process opens the way for its computational simulation, which in turns allows one not only to see whether the description matches the behaviour of the true system that is known *a priori*, but also to produce predictions about how it would behave under unseen circumstances or in the presence of various perturbations.

Models range from simple mathematical descriptions, relating the input and output of systems, to more mechanistic descriptions which attempt to capture the underlying mechanics of the system's workings. Naturally, the latter kind of models are generally able to provide more insight, but the increased level of detail entails a cost for both specifying the model (more things to tune) and analysing it. It is with the goal of facilitating this task that various modelling frameworks have been proposed. Several of them follow the *formal modelling* approach, whereby the description is underpinned by a logic interpretation, providing rigour to the model and enabling various analyses.

A particularly interesting class of models concerns dynamical systems, whose state evolves over time. This category of models encompasses many processes, both natural and artificial, in fields like electrical engineering, biology, physics and economics. While including dynamics in the description allows a model to express richer behaviour and hence lets one represent a greater variety of systems, the additional complexity also incurs a cost in the analysis of the model.

Recent developments in various scientific fields have seen a huge increase in the amount of data that is produced. On one hand, this change creates the need for new

methods of storing, accessing and analysing data. On the other, it opens the way for the development of more complex models, capturing an unprecedented level of detail. In practice, however, the difficulties of specifying and parameterising such complex models limit the size and complexity of the descriptions used. It is evident that, in order to make full use of the data, both modelling frameworks and statistical tools have to adapt so as to offer flexibility and efficiency.

A key issue when creating a model is to tune its parameters so that its behaviour is consistent with that of the true system. This step, although necessary, is nevertheless still far from solved in a satisfactory standard way. For simple mathematical models, established statistical methods such as least squares regression exist; as the complexity grows, however, so does the difficulty of this task. One of the main contributions of the field of machine learning has been to provide efficient solutions to this problem through leveraging statistics and probability theory and combining ideas originating from computer science and physics, among other fields.

Despite the contributions of machine learning, however, it is an all-too-common difficulty that solutions are developed in an ad-hoc manner tailored to specific problems. The lack of a standard representation for systems means that problems are being solved again and again with solutions being repeatedly developed from scratch. Additionally, the algorithms proposed are not aimed at non-experts, and are thus difficult to use to those who would gain the most from them — those working in the fields where the data is generated. This situation constitutes a major obstacle that is impeding the adoption of machine learning techniques in various fields.

With this in mind, there is a growing trend towards frameworks that allow the automated application of sophisticated algorithms, such as the Automatic Statistician [78], a web interface for identifying trends in temporal data and creating a human-readable textual description of them. Another example in this direction is the ever-growing family of *probabilistic programming* languages, which allow the specification of probabilistic models through a high-level syntax, and enable a non-expert to use inference algorithms without needing to delve into the details of their implementation. In spite of their benefits, such languages are still inadequate when it comes to their treatment of dynamical systems.

Another trend is the increasing recognition of the importance of randomness and uncertainty, which has implications in at least two distinct dimensions. Firstly, in the behaviour of systems, where non-determinism is now recognised as an important aspect which can critically affect the behaviour displayed. Here, the category of *stochastic*

systems is particularly important, as a widely-used framework for various dynamical non-deterministic processes. Stochastic systems have been extensively studied in both the formal modelling community and the machine learning literature; with respect to the latter, several approaches have been proposed, often involving various kinds of approximations due to the complexity of the dynamics.

A second implication of the significance of uncertainty appears in the context of inference: at the simplest level, systems are not always measured with perfect accuracy, and inference methods must properly account for such noise. Furthermore, when it comes to estimating parameter values, it is important to recognise that there is more information to be gained by also considering the confidence in the estimate instead of simply using a single value. This has favoured the use of the *Bayesian* methodology, which takes into account uncertainty about parameter estimates when making predictions.

When taken together, these factors indicate the need for a framework for the description and inference of stochastic systems which fulfils several different criteria: It must provide an accessible language that allows the description of large, complex systems in a concise and scalable way; this description must allow the inclusion of uncertain quantities, representing those aspects of the system about which our knowledge is incomplete; it must permit knowledge about observed behaviour of the system to be incorporated so as to update our initial beliefs in a statistically rigorous way.

Existing languages and frameworks do not satisfy these requirements. Starting from this point of view, in this thesis we develop a framework that is appropriate for the task as described above. We adopt a formal approach that offers not only a high-level language for specifying models, but also a way to reason about the behaviour of the system. In the process, we develop inference algorithms that are particularly suited for this setting, while also including algorithms proposed in the machine learning literature, adapted as appropriate.

Beyond the practical applications, the problem also holds theoretical interest: so far, formal frameworks have dealt with uncertainty or randomness in system behaviour in the form of non-determinism or stochasticity, but have not considered the uncertainty *about* this randomness. This is an important aspect of our knowledge about real-world systems, but incorporating it into formal models requires new mathematical objects and theory. In doing so, we present one of the first concerted efforts to bring together two fields, those of formal modelling and machine learning, which have so far existed mostly in parallel, treating the same kind of system but from different vantage points

and with different goals. We believe that this is a first step in what could be a very productive exchange of ideas and practices.

1.1 Contributions

This thesis explores how formal frameworks can be extended to cover uncertainty in system description, as well as what machine learning algorithms are appropriate for performing parameter inference in this setting. More concretely, its main contributions can be summarised as follows:

- We present a new formal language for describing stochastic systems which can capture uncertainty about parameter values and observed behaviour. This is the first application of the probabilistic programming paradigm to a high-level, system description language. At the same time, it extends the applicability of process algebras to systems with uncertain specification by introducing an appropriate underlying mathematical object. A preliminary version of this work has appeared in [40].

- We propose novel algorithms for inferring the parameters of Continuous Time Markov Chains, particularly suited for systems with infinite state-spaces, where many previously proposed methods were inapplicable, while achieving better performance than other similar approaches. This work has been published as [42].

- We develop a software framework that brings together the above elements, covering the tasks of parsing and analysing models as well as performing automated inference for them using a variety of algorithms. The framework is demonstrated on an epidemiological model using real-world data.

1.2 Structure of the thesis

This thesis is organised as outlined below:

- Chapter 2 gives the necessary mathematical background and contains an overview of relevant literature, including stochastic systems, formal modelling and machine learning with a particular emphasis on previous solutions for inference in stochastic systems.

- In Chapter 3, we present ProPPA, the language developed and its formal semantics. The chapter includes examples of inference results using simple algorithms.
- Chapter 4 presents algorithms for inference of stochastic systems with infinite state-spaces, based on a random truncation strategy, and an empirical analysis of their performance against state-of-the-art methods.
- Chapter 5 describes the implementation of the ProPPA framework, including the algorithms that form the core inference engine, as well as a case study showcasing the framework.
- Chapter 6 closes the thesis with a summary of its contents and our conclusions, and proposes further directions in which the work can be extended in the future.

Chapter 2

Background

This chapter offers an overview of the research context of the work presented in the thesis. We start with a brief presentation of the mathematical foundations of Markovian systems, before moving to formal languages for their description, with an emphasis on how uncertainty is incorporated. The focus then shifts to statistical techniques for learning unknown parameters of such systems, with a survey of different approaches taken in the literature.

2.1 Stochastic systems

A stochastic process X is a collection of random variables over a set $\mathcal{S}$ indexed by variable t: the indexing variable is often thought of as a time, although other interpretations are possible. We will write $X(t)$ for the instance of the process at time t. The time variable can be either discrete (e.g. integer-valued) or continuous and, similarly, the set $\mathcal{S}$ can be countable or uncountable.

Stochastic processes are used to describe systems which evolve in time in a non-deterministic way; at any time, there are generally multiple potential configurations, or *states*, of the system, represented as elements of the set $\mathcal{S}$, also called the *state-space* of the process. The distribution of $X(t)$ expresses the probabilities of these states at a particular point in time. The states are also viewed and referred to as the possible values of the process at any time. If $\mathcal{S}$ is uncountable, then $X(t)$ has an associated probability density function (pdf); if, instead, it is countable, $X(t)$ has a probability mass function.

There is a vast literature on the use of stochastic processes in physical sciences, engineering and social sciences as models for different activities and processes. A particularly important class is that of *Markov processes* (as defined in, for example, [36]).

Definition 1. A stochastic process $X(t)$ is a Markov process if it satisfies the following *Markov property*: for any number of time points $t_1 < t_2 < \ldots < t_n$ and states $x_1, \ldots, x_n \in \mathcal{S}$,

$$p\left(X(t_n) = x_n \mid X(t_1) = x_1, X(t_2) = x_2, \ldots, X(t_{n-1}) = x_{n-1}\right) = p\left(X(t_n) = x_n \mid X(t_{n-1}) = x_{n-1}\right) \tag{2.1}$$

This definition applies to both discrete processes, where p is a distribution, and continuous ones, in which case p is a pdf. Intuitively, the property means that the evolution of a Markov process depends only on its current state and not on its past. These processes are therefore also sometimes described as *memoryless*. A simple case occurs when we consider processes indexed by a discrete time variable and with a discrete state-space:

Definition 2. A Discrete Time Markov Chain (DTMC) is a tuple $\langle \mathcal{S}, \mathbf{P}, \pi_0 \rangle$ where:

- $\mathcal{S}$ is a set of states,
- $\mathbf{P}$ is a $|\mathcal{S}| \times |\mathcal{S}|$ matrix of transition probabilities such that each row sums to 1, and
- π_0 is an initial probability distribution over $\mathcal{S}$.

The transition probability matrix $\mathbf{P}$ of a DTMC controls the behaviour of the process: if at time n the state is i, then the i-th row of $\mathbf{P}$ gives the probabilities of the next state. In other words, $P(X(n+1) = j \mid X(n) = i) = P_{ij}$.

In this work, we are mostly concerned with processes indexed by a continuous time variable instead. A discrete-state, continuous-time process which satisfies the Markov property is called a Continuous Time Markov Chain; in the statistical community, such processes are more commonly known as Markov Jump Processes (MJPs). We will only consider *regular* or *non-explosive* chains, i.e. ones in which the number of transitions that occur in a finite time is finite with probability 1. Such processes admit a characterisation as in the following definition.

Definition 3. A Continuous Time Markov Chain (CTMC) is a tuple $\langle \mathcal{S}, \mathbf{A}, \pi_0 \rangle$ where:

- $\mathcal{S}$ is a countable set of states,
- $\mathbf{A}$ is the (infinitesimal) generator matrix, a $|\mathcal{S}| \times |\mathcal{S}|$ matrix whose rows each sum to 0, and
- π_0 is an initial probability distribution over $\mathcal{S}$.

The dynamics of CTMCs are simple to describe: the process can transition (jump) from one state to another, with these jumps taking place at random times governed by *rates*. These rates are connected to the matrix $\mathbf{A}$ as will be described below. If $r_{ss'}$ is the rate of jumping from state s to state s', then the probability of such a jump occurring in a small time interval $\mathrm{d}t$ is $r_{ss'}\mathrm{d}t$. The evolution of the probability of being at state $s \in \mathcal{S}$ must therefore satisfy the following ordinary differential equation (ODE):

$$\frac{\mathrm{d}\pi_t(s)}{\mathrm{d}t} = \sum_{s' \neq s} r_{s's}\pi_t(s') - \sum_{s' \neq s} r_{ss'} \cdot \pi_t(s) \tag{2.2}$$

The system of these ODEs, one for each possible state, is known as the *Chemical Master Equation* (CME), sometimes simply called the Master Equation, or the Chapman-Kolmogorov equation. Its solution gives the distribution over the state-space $\mathcal{S}$ at a specified time. If $\mathcal{S}$ is finite, then it can be shown that (2.2) admits the solution:

$$\pi_t = \pi_0 \, e^{\mathbf{A}t} \tag{2.3}$$

where π_t is a row-vector representing the distribution over all states at time t (i.e. the single-time marginal of the process), also called the *transient probability*, π_0 is the initial distribution of the process, and $\mathbf{A}$ is the infinitesimal generator. This matrix contains the transition rates between the states: specifically, for $i \neq j$, the entry $A_{ij} = r_{ij}$, the rate of jumping from state i to state j. The diagonal entries obey $A_{ii} = -\sum_{j \neq i} r_{ij}$, so that $|A_{ii}| = -A_{ii}$ is the total rate of leaving state i, called its *exit rate*. It follows that the sum of each row of $\mathbf{A}$ is zero.

Note that the quantity $e^{\mathbf{A}t}$ in Equation (2.3) is a *matrix exponential*:

$$e^{\mathbf{A}t} = \sum_{n=0}^{\infty} \frac{\mathbf{A}^n t^n}{n!} = I + \mathbf{A}t + \frac{\mathbf{A}^2 t^2}{2} + \dots$$

While simple to define, this quantity cannot be directly computed analytically. Several approximation approaches have been proposed (see [92] for a review) but it is difficult to quantify their performance, in addition to ensuring that they work well for a wide range of matrices. For example, methods based on decomposing $\mathbf{A}t$ can theoretically perform well, with a complexity similar to matrix multiplication. However, they become inaccurate when the matrix is ill-conditioned and cannot be used when it does not have a full set of linearly independent eigenvectors. It is also worth noting the existence of techniques such as in [2] which are aimed directly at approximating $\pi_0 \, e^{\mathbf{A}t}$, and can achieve better performance by considering this more limited problem. In summary, while (2.3) in principle gives a way to compute the distribution of a (finite) CTMC at any

point in time, it is not always useful in practice due to the difficulty of exponentiating large matrices.

The above analysis makes the assumption that the transition rates remain constant. A CTMC for which this is true is called *homogeneous*. Letting the rates change with time allows for capturing richer behaviours, but makes analysis of the resulting model more complex. Hereafter, we only consider homogeneous CTMCs without making this explicit.

There exists an equivalent way of describing a CTMC. As implied by the Markov property, the time taken to jump from state i to state j must follow an exponential distribution with rate r_{ij}, i.e. with mean $\frac{1}{r_{ij}}$. Exploiting properties of the exponential distribution, it can be shown that the total time spent in state i (i.e. the time before any jump occurs) is also exponentially distributed, with rate $r_i = \sum_{j \neq i} r_{ij}$; this is the same as the exit rate seen above as the absolute value of the diagonal terms in the infinitesimal generator matrix. States to which transitions are more frequent (i.e. for which r_{ij} is high) have a higher probability of being the target of a transition. In other words, for a given starting state i, every other state $j \neq i$ has a transition probability $p_{ij} = \frac{r_{ij}}{r_i}$ that expresses how likely j is to be the next state once a transition occurs.

This parameterisation gives rise to a method of simulating a CTMC, that is, of drawing a sample path from the distribution it defines. Reported by Gillespie in [46], it is known as the Stochastic Simulation Algorithm (SSA) and was first motivated by simulating the reactions in chemical mixtures, but is generally applicable to any CTMC. The basic algorithm is shown as Algorithm 1. Various optimisations and approximations have been proposed to improve the running time or memory requirements, such as the τ-leaping method [47]. Alternatives to the SSA also exist, including Gibson and Bruck's next reaction method [44]. We will not go into more detail about the differences and relative benefits of these, as they are not of particular interest for the work in this thesis. It is enough to stress that simulating a CTMC is a well-researched problem with solutions that are established as standard, albeit one that is still attracting attention due to computational difficulties and special cases. On the other hand, calculation of the transient probabilities is a much more difficult question, and more central to this work, as will be explained later in this chapter.

CTMCs are often used to describe systems of interacting components. If all components of a certain type are indistinguishable from each other, we may only be interested in the *number* of instances of each type. The state of the system can then be represented as a vector containing these counts. Such a process has state-space $\mathcal{S} \subseteq \mathbb{N}^M$, where

Algorithm 1 Gillespie's Stochastic Simulation Algorithm

Input: CTMC description as rates r_{xy}, initial state s_0, final time t_f

$s \leftarrow s_0$
$t \leftarrow 0$
repeat
    Compute rates of all transitions $r_{ss'}$ from s, and exit rate $r_s = \sum r_{ss'}$
    Sample waiting time $\Delta t \sim \text{Exponential}(r_s)$
    Choose next state s' with probability $\frac{r_{ss'}}{r_s}$
    Jump: $s \leftarrow s'$, $t \leftarrow t + \Delta t$
until $t \geq t_f$

M is the number of different types of components, and is called a population CTMC (pCTMC) or population MJP (pMJP). An important issue which limits the analysis of pCTMCs is the so-called *combinatorial explosion*: if there can be between 0 and N instances of each component, the total number of states $(N+1)^M$, i.e. the size of the state-space grows exponentially with the number of components. This means that naive solutions can quickly become inapplicable for non-trivial systems.

2.2 Formal Modelling and Process Algebras

While CTMCs are used as models for many physical, social or engineering processes, they can be difficult to manage as they grow in size. Writing down the process explicitly is cumbersome, repetitive and prone to introducing errors. This has led to the development of high-level languages for the description of CTMC-based systems. Equipped with a precise mathematical semantics, these formal frameworks allow both an easier specification of processes and the automated analysis of systems.

One example of such approaches is the *rule-based* paradigm of modelling languages like Kappa [25], where the possible transitions in the system are described as rules. Each rule corresponds to a kind of interaction: it specifies an input pattern, possibly matching multiple states, and how that pattern is transformed by the transition. The resulting description style is closely related to graph rewriting. *Petri Nets* [104] are another formalism, primarily used for visualising and simulating systems involving various kinds of communication and computation. This work focuses on a particular class of modelling languages called *process algebras*.

Process algebras (or process calculi) are languages whose formal nature allows

rigorous analysis of systems and verification of their properties. The field dates back to Milner's Calculus of Communicating Systems (CCS, [90]) and Hoare's calculus of Communicating Sequential Processes (CSP, [59]), languages introduced to describe concurrent computation. The behaviour of the various agents involved in the system is described by terms conforming to a formal syntax. These include references to the different actions that can occur, which model, for example, computation, synchronisation or change of state. Behaviour patterns like communication between agents, parallel composition, alternative choices can be expressed in a concise form which admits automatic mathematical treatment.

An advantage of this formal character is that one can reason about the behaviour of the system simply by analysing its syntactical description. For instance, it is possible to check whether the system can become deadlocked or is guaranteed not to. It is further possible to prove whether two processes behave in the same way and can be considered equivalent. Moreover, the languages and their analysis is compositional: after specifying the individual components, the behaviour of the whole system arises naturally from them and from the way they are composed using the language's operators. A further benefit of formality is that complex properties can be expressed in an appropriate logic, and the behaviour of the system can be formally analysed to verify whether it satisfies a given property, via *model-checking* algorithms.

Languages like CCS and CSP provide qualitative description of systems: what configurations are possible and how the system can get there. They can capture non-deterministic behaviour, but contain no description of timing in the system. Hillston's Performance Evaluation Process Algebra (PEPA, [58]) was one of the first to include quantitative information, making it an early example of a Stochastic Process Algebra (SPAs, [55]), a sub-family in which actions have associated delays. In PEPA, the duration of an action is assumed to follow an exponential distribution, making it a suitable language for describing CTMCs. Some SPAs follow this assumption; others differ, allowing other timing models that correspond to non-Markovian processes. The inclusion of quantitative aspects permitted PEPA, and SPAs in general, to be used to analyse the performance of, for example, computer systems, by describing large models, simulating them and extracting measures of interest such as the throughput or expected uptime.

As mentioned above, a process algebra model follows a formal syntax specified using a grammar. The various terms, corresponding to parts of the system, are given formal meaning via the definition of a *semantics*. This normally maps a term to

a Labelled Transition System or some other mathematical object, which concretely describes the behaviour of that term.

Definition 4. A Labelled Transition System (LTS) is a triple $\langle V, \mathcal{L}, \delta \rangle$, where

- V is a set of states
- $\mathcal{L}$ is a set of labels
- $\delta \subseteq V \times \mathcal{L} \times V$ is a transition relation between the vertices, labelled with an element of $\mathcal{L}$.

An element $(v, l, v') \in \delta$ is usually written as $v \xrightarrow{l} v'$, which indicates that the label for transitioning from v to v' is l. The label gives additional information about the transition. In a SPA, for instance, it can include the corresponding rate.

In the case of PEPA and other languages with exponentially-distributed action times, the model can also be seen as specifying a CTMC. The semantics also describes how to construct a CTMC from a given model, thus giving it a direct formal interpretation in terms of a stochastic process and enabling further analysis.

Numerous SPAs have been developed, promoting different modelling styles or offering features adapted to specific cases. Despite their origins in performance modelling, the field of biology (and, in particular, systems biology) has also provided fertile ground for further development, both in offering applications (systems to be modelled) and in motivating the development of new languages, either specialised ones or general-purpose but with features inspired by or more suited to biological modelling. The latter category includes SPAs like Beta-Binders [108] and Bio-PEPA [23], a language with a similar syntax and semantics to PEPA but with features that make it more suited to biological modelling. Bio-PEPA adopts some biochemical terminology, such as referring to the types of components and interactions that appear in the system as species and reactions, instead. A model can be interpreted as a pCTMC or as a set of ODEs — the latter being an example of so-termed *fluid* analysis of formal models that has been receiving attention recently, where the state of the system is viewed as continuous rather than discrete (see also Section 2.6.4).

2.3 Description of systems with uncertainty

The formalisms mentioned above assume a full specification of the system is known, yet it is very rarely the case that one is available. Instead, when modelling a real system, one

often has incomplete knowledge about it. Additionally, concrete examples of behaviour or measurements of the system may be available. When creating a model of a physical or computer system, for instance, while the different kinds of components involved and the ways in which they can interact may be known, the precise details of these interactions (such as the rates of various events) are generally not. One may have some idea about these unknown quantities, such as the likely range of their values, obtained, for example, from existing expert knowledge of the field. A similar situation arises when trying to construct a system that fulfils certain properties. In both cases, the uncertain quantities of the model must be tuned so as to either match the observed behaviour or satisfy the requirements. This section discusses how this uncertainty has been considered in the literature, both in terms of providing an adequate underlying mathematical object, and in methods for resolving uncertainty in process algebra models.

2.3.1 Markov Chain generalisations

Perhaps due to their complexity, the question of generalising CTMCs has not been directly considered much. Instead, most relevant work has focused on DTMCs.

A first extension of "conventional" Markov Chains was provided in [64] in the form of Interval Markov Chains (IMCs). An IMC is a generalisation of a DTMC where the transition probabilities are specified via intervals rather than specific values. In this case, giving an interval means that the value of the probability cannot be provided more accurately, but is still assumed constant. This provided a simple way of capturing uncertainty.

A slightly different interpretation for the specification of probabilities as intervals was taken in [118]. There, a variant of IMCs is proposed where the values of the transition probabilities can change: the jump probabilities from state s are resampled from the corresponding interval whenever s is visited. The authors call this an Interval Markov Decision Process, and contrast it to the so-called Uncertain Markov Chain semantics of [64]. They consider the problem of verifying whether a system satisfies a certain property and derive complexity bounds for it under each interpretation. It is also worth noting that, in both [64] and [118], there is no implicit assumption that a probability is equally likely to lie anywhere in its interval; no consideration is given to the distribution of values, beyond upper and lower bounds.

A similar framework is presented in the form of Abstract Markov Chains [33], although from a different angle. These are, again, DTMCs where the transitions are

labelled with intervals rather than single probabilities. An extension of this to the continuous-time domain has been given in [67]. In both cases, the focus is on using the interval-valued versions as abstractions for concrete systems: a state of the abstract chain can correspond to multiple states of the concrete one, necessitating the use of intervals to capture the range of transition probabilities or rates from all those concrete states. Model-checking these abstractions is also considered, and is achieved by using a three-valued logic where a property can be evaluated as true, false or "don't know".

A related matter is also treated in [72]. That work extends the calculation of transient probabilities in a DTMC when the jump probabilities are given as intervals. In other words, the authors show how to find an interval in which the transient probability is guaranteed to lie for any values of the jump probabilities that satisfy the interval constraints. This could easily be applied to IMCs in order to provide a transient solution calculation, although the connection is not made in that work. Very recent work [15] studied pCTMCs with incomplete specifications, with a focus on deriving efficient methods for approximating their behaviour in the limit of large populations. The authors distinguish between two scenarios, based on whether the parameters controlling the unknown transition rates are fixed or can change in time (similar to the distinction between Uncertain Markov Chain and Interval Markov Decision Process semantics outlined above).

While useful as a first step, IMCs are limited in their descriptive power, since they require the possible values for a particular probability to lie in an interval. As was shown by Caillaud *et al.* in [18], they are narrow enough to not be closed under operations such as conjunction — indeed, it is simple to construct a system by combining two IMCs such that the acceptable values for the probabilities do not form intervals. To deal with that, Caillaud *et al.* introduced *Constraint Markov Chains* (CMCs). These are, again, generalisations of DTMCs, in which the acceptable values for the transition probabilities do not necessarily form an interval but can lie in any set. CMCs are general enough to be closed under conjunction and parallel composition. The full definition, which we will make use of later in Chapter 3, is given below as it appears in [18]:

Definition 5. A Constraint Markov Chain is a tuple $\langle S, o, A, V, \phi \rangle$, where:

- $S = \{1, 2, \ldots, k\}$ is the set of states
- $o \in S$ is the initial state
- A is a set of atomic propositions

- $V\colon S \to 2^{2^A}$ gives a set of acceptable labellings for each state
- $\phi\colon S \times [0,1]^k \to \{0,1\}$ is the *constraint function*, such that if $\phi(i,\mathbf{x}) = 1$ then $\mathbf{x} = (x_1,\ldots,x_k)$ is a vector of probabilities, i.e. $\sum_{j=1}^{k} x_j = 1$

The binary-valued constraint function ϕ formalizes which values of the transition probabilities are accepted: $\phi(s,\mathbf{x}) = 1$ iff $\mathbf{x}$ is an acceptable vector of transition probabilities from state s, i.e. it both defines a valid probability distribution over target states and satisfies the constraints of the particular model. The function V assigns to each state a set of sets of propositions; this means that there are potentially many different possible labellings for each state, representing possibly incomplete or uncertain knowledge about the properties that hold in each state. Once again, no distribution is assumed over the acceptable values of the probabilities.

A similar idea had previously appeared in [9], although the development of CMCs seems to have been independent. The authors of that work consider Generalized Markov Processes, collections of DTMCs whose transition probabilities satisfy certain constraints. The language for specifying these constraints, while more general than simple IMCs, allows for less flexibility than what CMCs afford.

2.3.2 Optimisation of formal models

A different direction of research has dealt not with incorporating uncertainty into models but rather finding parameterisations of models that satisfy some criteria. In other words, the workflow starts with expressing the model in the language of interest, perhaps with some arbitrary values for the unknown parameters, and specifying a criterion that expresses goodness of fit (for example, how well the expected output of the system matches some data). The work then proceeds by finding good parameter values that maximise the criterion.

Optimization of formal models is usually performed manually as a separate step on a case-by-case basis, and can be a time-consuming and error-prone process. There has been some recent work into automating the task of finding optimal parameters for models in specific languages; given the difficulty of the problem, these mostly use heuristics to manage the search space. The Evolving Process Algebra framework [81, 82] works with descriptions of systems in PEPA or Bio-PEPA and uses genetic algorithms to find good values for parameters. The usual genetic computation procedure is to start with some randomly chosen parameterisations and then repeatedly construct a new "generation" of them by permuting and combining existing ones; parameter values that

score high on some fitness measure are likelier to be selected for the next generation. In this case, the fitness of a parameterisation is measured by simulating the system for those values multiple times, averaging the resulting traces, and taking the Euclidean distance between this average and the observed time-series. The work in [117] looks at parameter optimisation for models written in the Calculus of Wrapped Components, a language particularly suited to describing biological systems with locations. This approach searches for a good solution using constraint optimization methods, claimed to be less computationally intensive than evolutionary computation approaches like genetic algorithms. In a similar spirit, the capacity planning tool of [131] automatically finds parameters for PEPA models that minimize a specified cost function while obeying desired performance requirements. In this case, the parameters searched for are populations of the various components. The search is performed using Particle Swarm Optimization techniques. Recent work by Li *et al.* [76] has examined how to find parameters that make a Markov Chain match a given behaviour expressed in a temporal logic. The optimisation problem is reduced to a set of equations and solved using a theorem prover. This approach can also be applied to the optimisation of IMCs.

In the above frameworks, the optimisation step is orthogonal to the description: the parameters are assumed to have a single value which is unknown, and the goal is to find potential good values for them. A major drawback is that the methods return a single value (or a collection of good fits) without any quantification of the uncertainty in them. The lack of a full probabilistic model behind the optimisation limits how informative the results are.

Another, more complicated, task is that of synthesising a whole model based on some observed data. This involves not only the numerical parameters but also the structure of the model itself. This has been investigated for simple examples expressed in a subset of the stochastic π-calculus in [112] . A subsequent version of the Evolving Process Algebra framework also includes this capability for Bio-PEPA models by exploring different possible species definitions [83] and expressions for the rates [99]. Both approaches work with evolutionary computation methods, by creating models that respect the syntax of the language and provide a good match to the observed behaviour.

More sophisticated inference methods have been explored (and will be described below), but they have generally been developed to work with an explicit mathematical description of the process, rather than a high-level model. A modeller is therefore generally faced with two options: use a high-level language but with limited learning capabilities; or forego the attractive features of formal languages in order to have access

to more powerful inference.

2.4 Probabilistic Programming

The issue of inference methods only being available for low-level representations of models is not particular to CTMCs. Indeed, in the machine learning community, there is a tendency to code models from scratch and develop bespoke inference solutions for them, which are then not straightforward to implement for other examples. This implies heavy duplication of code and lack of any framework or implementation that serves as a standard. The probabilistic programming paradigm, whose prominence has been increasing in recent years, attempts to reverse this trend by placing emphasis on writing the model instead of the underlying inference algorithm.

The BUGS system [120] could be considered as precursor or early example of this paradigm. Following in its footsteps, probabilistic programming languages offer a high-level, user-friendly language for specifying probabilistic models, and automate the process of inference. Some, like Church [50], were designed to be probabilistic extensions of existing languages, while others, like IBAL [105], simply borrowed syntactic and other elements from established languages. The Infer.NET framework is built as a library that can be called by various languages, but has also been used as an inference back-end for the Fun [14] and Tabular [52] interfaces to it. Many of the proposed languages follow a functional paradigm, perhaps for ease of semantical definitions and analysis. In recent years, various other probabilistic programming languages have been introduced, such as Anglican [132], Stan [21], R2 [98], WebPPL [51] and Venture [91]. The key advantage of all these frameworks is that the user can leverage the strengths of a full programming language in order to express the model of interest, provide observations and perform inference without needing to specify all the details of the underlying algorithm. Within this shared pattern, the various languages exhibit differences in the inference algorithms used, the modelling style and the kind of models that they favour or are more suited to describing. Recent work has also started exploring issues like decidability and complexity in this new paradigm [66].

Nevertheless, existing languages have been very limited in their treatment of dynamical systems. With no provision for time as a special variable, it is difficult (or outright impossible, in some cases) to describe systems with evolving state. Things may be easier when one is considering a process with a finite number of steps, but continuous time is hard to capture accurately. A recent survey of probabilistic programming lan-

guages [53] briefly addresses the issue: it suggests a rudimentary way of simulating continuous time with discrete-time models, but concludes that it is unsatisfactory on multiple levels, including performance. It therefore recommends a principled, explicit representation of time for continuous-time systems, but warns that this may introduce additional complexity both in the semantics of the language and in the cost of performing inference. The only language which caters to this requirement is CTPPL [106], the Continuous-Time Probabilistic Programming Language, of which there seems to be no available implementation or follow-up work after the initial theoretical presentation. Even so, while providing constructs for describing time and its effect, the specific dynamics of the system of interest still have to be described explicitly. This is not only a repetitive task (for example, all CTMCs essentially share the same underlying dynamical model, which would have to be provided separately each time) but it is sometimes disallowed by restrictions on the syntax of the language.

This limitation is a serious hurdle for the adoption of such frameworks in various scientific fields, where models like CTMCs are employed often. One could claim that these issues, at least to an extent, stem from how new languages are developed: most frameworks start by adapting the syntax or principles of an existing programming language. However, these languages lack the constructs for describing complex systems in a straightforward and concise way. This is precisely one of the reasons why higher-level formal languages were adopted — to facilitate the specification and analysis of systems that would otherwise be hard to manage. It makes sense, therefore, to also consider this modelling style in conjunction with the probabilistic programming paradigm.

A first step towards this is the ABC-Fun language [41], developed as part of this PhD project, which uses an extension of the Fun language to describe stochastic systems in simple notation. The language extensions meant that the Fun inference engine could not be used, and a simple approximate algorithm was employed for performing inference. While an interesting stepping stone, ABC-Fun left things to be desired. Firstly, the language itself is rather low level and does not easily lend itself to any formal analysis. Secondly, the inference was basic, due to the difficulties presented by the stochastic setting. ProPPA, the language described in this thesis, addresses both of these concerns by being rooted in the process algebra methodology and employing more sophisticated inference techniques.

Background information related to formal modelling has been given earlier in this chapter. The next section is an overview of relevant Machine Learning techniques,

followed by a survey of approaches proposed in the literature for inference of CTMC-based systems.

2.5 Learning

This section presents the basics of Bayesian inference and some standard techniques of statistical machine learning. For a fuller and more principled treatment, the reader is referred to [12] or other textbooks. We assume that the behaviour of a system of interest depends on a number of parameters, collected in a vector θ. Our goal is to learn these parameters based on some observed output D from the system. For ease of presentation, in the following we assume that the parameter is one-dimensional unless otherwise noted.

2.5.1 Bayesian statistics

Since the parameters are unknown, a natural approach that accounts for the uncertainty in their values is to represent them as random variables. One of the benefits this leads to is that we can express existing knowledge about the values of the parameters; this is encapsulated in a *prior* distribution $p(\theta)$, which expresses our belief about which parameter values are likely before any data is considered.

Once observations are taken into account, this belief will naturally shift to accommodate the data. Our goal is to compute this *posterior* distribution. We can view the observations and parameters as coming from some unknown joint distribution $p(\theta, x)$. Note that this formulation does not imply that the parameters and data have multiple values or are generated repeatedly; probability distributions are used to express uncertainty and "reasonable expectation", as elaborated on in [24]. In probability terms, the posterior is simply the conditional distribution of the parameters given the data, i.e. $p(\theta \mid D)$. An immediate application of the laws of probability then gives a way to compute the posterior in principle, in the form of *Bayes' Rule*:

$$p(\theta \mid D) = \frac{p(\theta)p(D \mid \theta)}{p(D)} = \frac{p(\theta)p(D \mid \theta)}{\int p(\theta')p(D \mid \theta')\mathrm{d}\theta'} \tag{2.4}$$

This means that the posterior can be computed exactly if all the quantities involved are known. Unfortunately, the *marginal likelihood* $p(D)$ is an integral over a potentially high-dimensional space and can be difficult or intractable to compute. Additionally, in some models the likelihood is also expensive or impossible to compute. In some

cases, the combination of the prior and likelihood permit the analytical extraction of the posterior without the full computation being necessary; such cases, however, are limited to specific distributions.

The Bayesian approach described above allows one to consider the impact of all possible values of a parameter when reasoning about the system. For instance, if the output of the system is given by a function $f(x;\theta)$ of its input x, then, for a given input, considering a distribution π over the parameters θ, whether prior or posterior, results in a corresponding distribution over possible outputs. This predictive distribution can be described in terms of confidence intervals or, if desired, can be "compressed" to a single value. One possibility for such a Bayesian-style point prediction is to choose

$$y = \mathbb{E}_\pi[f(x;\theta)] = \int f(x;\theta)\pi(\theta)\mathrm{d}\theta$$

that is, the expectation of the predictive distribution. This choice minimizes an expected loss of the form $\mathbb{E}\left[(y-y_0)^2\right]$ where y_0 is chosen from the predictive distribution. Other point predictions are possible, such as by using the median of the predictive distribution instead of its expectation, according to the loss function one is interested in minimising.

A simpler alternative is *maximum-likelihood* estimation, which attempts to find a single value for the parameter by maximizing the likelihood:

$$\theta_{ML} = \arg\max_\theta p(D \mid \theta) \tag{2.5}$$

In this case, predictions will only consider a single possible value of the parameter:

$$y_{ML} = f(x;\theta_{ML})$$

This can give very misleading results in cases where the probability mass is not sufficiently collected around the estimate θ_{ML}. It also bears noting that, despite being simpler, solving the necessary maximization problem (2.5) can still be very expensive for complex models. An additional concern, although perhaps less immediate in dynamical systems, is the risk of overfitting: finding a value which explains the observed data very well, but does not generalise well to unobserved inputs and therefore produces inaccurate predictions.

Overall, the Bayesian approach presents multiple benefits. It allows the incorporation of existing knowledge. Because it considers the whole distribution of parameters, it produces more rigorous estimates that are not prone to overfitting. By viewing unknown values as distributions, it also allows for a principled treatment of missing data, which can be represented as additional parameters. However, this comes at the price of

additional computations, some of which can be intractable. A large set of techniques has been introduced to deal with these difficulties, some of which are discussed in the following section.

2.5.2 Sampling methods

2.5.2.1 Markov Chain Monte Carlo

Many applications involve distributions whose pdf makes computations (such as in Equation (2.4)) and sampling from them difficult. In these cases, *Monte Carlo* methods are a standard tool for drawing samples from the distribution of interest p, which can then be used either to empirically reconstruct p or to compute quantities of interest. Specifically, if X is a random variable with pdf p, then the expectation of a function of X can be approximated as

$$\mathbb{E}[f(X)] = \int f(x)p(x)\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N} f(X_i) \tag{2.6}$$

where the samples X_i are drawn from p. This *Monte Carlo* integration becomes exact as $N \to \infty$; a finite number of samples introduces a bias in the approximation. The question therefore becomes how to efficiently draw samples from p. Among the various methods proposed for this purpose, this work makes use of the category of Markov Chain Monte Carlo (MCMC) algorithms: in these, the samples are not drawn independently from p but are instead correlated with each other. The methods are asymptotically exact: in the limit of infinite draws, the samples drawn are representative of p. However, estimating whether a finite number of steps is representative is a non-trivial task, as will be discussed in Section 2.5.2.3. The presentation in the following makes no assumptions about the distribution we are interested in sampling from, but can be directly applied to the posterior estimation setting, i.e. for sampling from $p(\theta \mid D)$. More details can be found in dedicated works like [77].

The simplest such method is the *Metropolis-Hastings* (M-H) algorithm. M-H is applicable if the target pdf $p(\theta)$ can be computed at every point of its domain. The method works by repeatedly proposing new values and accepting them probabilistically. Specifically, from any point θ, a move is proposed to θ' from a *proposal distribution* $q(\cdot \mid \theta)$. The move is then accepted with probability

$$a = \min\left(\frac{p(\theta')}{p(\theta)}\frac{q(\theta \mid \theta')}{q(\theta' \mid \theta)}, 1\right) \tag{2.7}$$

This process defines a random walk on the domain of interest, as sketched in Algorithm 2. The performance of the algorithm depends to a large degree on the choice of proposal q, which determines how large the steps in the search space are and, consequently, how likely we are to accept or reject a proposed value. In bad cases, the chain can become stuck in a value (because the proposed values have low probability under p) and fail to explore important parts of the search space. Choosing a good proposal distribution is particularly difficult in higher dimensions, where exploring the space becomes problematic. Theoretical analysis [39] has shown that, in higher dimensions and under certain regularity conditions on the target distribution, the M-H algorithm performs optimally when the acceptance rate is approximately 23%.

Algorithm 2 Metropolis-Hastings sampling

1: Sample θ from the support of p
2: $i \leftarrow 1$
3: **repeat**
4: Propose $\theta' \sim q(\cdot \mid \theta)$
5: Compute the acceptance ratio a from (2.7)
6: With probability a:
7: Accept θ' and set $\theta \leftarrow \theta'$
8: $\theta_i \leftarrow \theta$
9: $i \leftarrow i+1$
10: **until** enough samples are taken

The algorithm constructs a chain of θ_i (where it is possible to have $\theta_i = \theta_{i+1}$) and can be described by a transition kernel $k(\cdot,\cdot)$: if the current state of the chain is θ, the probability of the next state being θ' is $k(\theta,\theta')$. A distribution p^* is said to satisfy *detailed balance* with respect to the transition kernel k if

$$p^*(\theta)k(\theta,\theta') = p^*(\theta')k(\theta',\theta)$$

Detailed balance is a useful property because it guarantees that, if the chain eventually converges to an equilibrium distribution (i.e. if it is *ergodic*), then that distribution is p^*. Equation 2.7 ensures that the distribution p satisfies detailed balance for the chain of θ_i, and so the samples returned from Algorithm 2 will be drawn from the target distribution.

Note that the algorithm does not necessarily require the pdf of the target distribution itself to be computable, but only an unnormalised version of it. Indeed, assume that

$p(\theta)$ is not available but can be written as $p(\theta) = \frac{\gamma(\theta)}{K}$, where γ is a function we can compute. The acceptance ratio (2.7) can be rewritten as

$$a = \min\left(\frac{\gamma(\theta')}{\gamma(\theta)}\frac{q(\theta \mid \theta')}{q(\theta' \mid \theta)}, 1\right)$$

This is very useful for Bayesian parameter inference, where the target distribution is the posterior $p(\theta \mid D)$, whose computation is in general not tractable due to the term $p(D)$. Using (2.4), however, we can apply M-H sampling with $\gamma(\theta) = p(\theta)p(D \mid \theta)$.

A special version of the M-H can prove to be more efficient in certain cases, where there are at least two dimensions to be sampled from. When the parameter has more than one dimension, we can update its value in each dimension in turn. Under the *Gibbs sampling* scheme, we propose new values from the conditional distribution of each parameter given all the others. For example, for a two-dimensional parameter $\theta = (\theta_1, \theta_2)$, we choose to first update θ_1 by proposing from $q_1(\theta_1' \mid (\theta_1, \theta_2)) = p(\theta_1 \mid \theta_2)$ (similarly, we propose a new θ_2' from $p(\theta_2 \mid \theta_1)$). It is easy to show that, with this choice, the acceptance ratio in (2.7) becomes $\alpha = 1$, i.e. a sample is always accepted. This reduces the random walk behaviour, achieving faster convergence, and eliminates the need to construct a proposal distribution. However, the method is only applicable when the conditional distributions can be sampled from (note that this does not require their densities to be known or computable, since the acceptance ratio does not need to be computed).

The introduction of intermediate auxiliary variables, while counterintuitive, can sometimes facilitate the formulation of a Gibbs sampling scheme by allowing one to sample from the necessary conditional distributions. Importantly, the samples returned come from the joint distribution of all the variables. Therefore, considering only some of the dimensions gives results that follow the marginal distribution of the variables of interest.

2.5.2.2 Pseudo-marginal methods

In many applications, the distributions to be sampled from are too complex to allow for calculation of their density (thereby making M-H schemes ineffective or inapplicable) and do not exhibit a structure that makes conditional pdfs available (disallowing Gibbs sampling). Sometimes, however, it is possible to calculate an estimate $\hat{p}(\theta)$ of the target density through some random process. The work by Beaumont [10] and Andrieu & Roberts [8] showed that such estimates can be used in a M-H sampler instead of the correct density p without affecting the target distribution of the chain. This is

achieved as long as the estimator $\hat{p}$ is unbiased, i.e. $\mathbb{E}[\hat{p}(\theta)] = p(\theta)$. These so-called *pseudo-marginal* methods have been characterised as "exact-approximate": the samples drawn still come from the distribution of interest, despite using an approximation to its density. Their use allows the application of MCMC methods to a wider range of problems.

The validity of this scheme can be explained simply. The pseudo-marginal M-H algorithm essentially replaces the acceptance ratio of (2.7) with

$$a = \min\left(\frac{\hat{p}(\theta')}{\hat{p}(\theta)}\frac{q(\theta \mid \theta')}{q(\theta' \mid \theta)}, 1\right) \tag{2.8}$$

The calculation of the estimator $\hat{p}$ involves some random quantities u, which will change between different estimates. We can consider these u as additional random variables to sample from. The pseudo-marginal M-H algorithm can then be viewed as a simple M-H walk over an augmented space, consisting of both the variable of interest θ and the random variables u. New states for this chain are proposed in two steps: first a new value θ' is sampled, then a new u' is generated from some (possibly unknown) distribution $\pi(u')$. It can be shown that, when using (2.8), the target distribution of the chain is $\hat{p}(\theta \mid u)\pi(u)$. The key point is that the marginal of this for θ is

$$\int \hat{p}(\theta \mid u)\pi(u)\mathrm{d}u = p(\theta)$$

since $\hat{p}$ is an unbiased estimator. Hence, the values of θ are drawn from the correct distribution. Note that for the scheme to work it is important that $\hat{p}(\theta)$, once computed, is used in all subsequent calculations of the acceptance ratio as long as the state of the chain remains θ. Essentially, this means "caching" the estimated value and not recomputing it.

Although the only requirement is for the estimate $\hat{p}$ to be correct on average, recent work (e.g. [119]) has shown that its variance can significantly affect the performance, in particular hindering convergence of the sampling chain. It is therefore desirable to construct an estimator that is not only unbiased, but also low-variance, to make the method more robust. The construction of such estimators and the analysis of the effect of the variance is an active area of research. Another concern is ensuring that the estimator does not produce a negative value, which would be problematic when used in the place of a probability, such as in the acceptance ratio of a M-H step. McLeish [87] and Rhee & Glynn [111] examine a general scheme for "de-biasing" estimators, that is, obtaining unbiased estimates from biased ones. Agapiou *et al.* [1] consider different ways of removing the bias from estimates obtained by MCMC methods, particularly

focusing on infinite spaces. Jacob & Thiery [61] examine the theoretical existence of estimators that are both unbiased and guaranteed to be non-negative under different generation schemes. Murray & Graham [96] propose a way of using slice sampling, another MCMC algorithm, to improve the performance of pseudo-marginal methods.

2.5.2.3 Monitoring convergence

MCMC methods do not immediately generate samples from the target distribution. At first, the samples taken depend heavily on the initialisation of the chain. Eventually, the proposal and acceptance dynamics ensure that the initial state is "forgotten" and that the chain has converged to its target distribution. Samples taken after this point are representative of the distribution and can be used as in (2.6); previous samples, however, ought to be discarded as *burn-in* for the chain.

Unfortunately, determining whether the chain has converged is not a trivial task, and there are no known faultless methods for deciding whether the burn-in stage is complete; instead, heuristics are employed. The most common approach relies on calculation of the Potential Scale Reduction Factor (PSRF), a metric that requires running multiple chains from different initial values. We give a brief description of how this statistic is computed; more details can be found in [38]. Assume that we have run C chains for N steps each, and let $\theta_j^{(i)}$ indicate the i-th sample from the j-th chain. Let $\overline{\theta_j}$ be the mean of the samples from the j-th chain, and $\overline{\overline{\theta}}$ be the mean of all samples. The variance between the samples of each chain can be computed as

$$V_j = \frac{1}{N-1} \sum_{i=1}^{N} \left(\theta_j^{(i)} - \overline{\theta_j} \right)^2$$

while the variance across different chains is

$$W = \frac{N}{C-1} \sum_{j=1}^{C} \left(\overline{\theta_j} - \overline{\overline{\theta}} \right)^2$$

An overall variance is then estimated via

$$Var = \left(1 - \frac{1}{N} \right) W + \frac{1}{N} \frac{1}{C} \sum_{j=1}^{C} V_j$$

The PSRF is defined as

$$PSRF = \sqrt{\frac{Var}{W}} \tag{2.9}$$

The chains are considered to have converged if $PSRF < 1.01$. Essentially, we require the within-chain variance to be close to the between-chain variance, indicating that the

samples in different chains are generated from similar distributions. When the search space has more than one dimension, a value of the PSRF can be computed by this process for each dimension. Although useful, the PSRF may be a misleading metric in some cases. For example, if the target distribution is multimodal, it is possible that each chain is sampling correctly from one part of the distribution. When combined, the samples from all chains would accurately represent the whole distribution, but the PSRF calculation may not indicate that.

The samples returned by an MCMC algorithm are correlated with each other, and hence provide less information than the same number of independent samples would. Another useful statistic to consider is therefore the Effective Sample Size (ESS) of the output of a chain. This gives a measure of the "equivalent" number of independent samples. It is defined as

$$ESS = \frac{N}{1 + 2\sum_{k=1}^{\infty} \rho_k} \tag{2.10}$$

where N is the total number of samples and ρ_k is the autocorrelation of the samples at lag k. This definition assumes an infinite-length sequence of samples and is therefore theoretical. For the experiments in this thesis, the ESS was estimated by stopping the sum in Equation (2.10) early, following the *initial positive sequence estimator* of [43].

2.5.3 The Forward-Backward algorithm

Dynamical systems arise often in machine learning, and one of the standard tools for handling them is the so-called Forward-Backward algorithm. Consider a discrete-time system with finite state-space, such as a DTMC. Let p_{ij} be the known probability of transitioning from state i to state j, and assume that at any time n, the state x_n is observed through a noise model $\pi(y_n \mid x_n)$. The goal is to infer the state of the system at different observation times, i.e. compute $p(x_n \mid y_1, \ldots, y_N)$ for $n = 1, \ldots, N$. The general idea is to separate the effect of the observations prior to the time of interest from those that come after it. This algorithm therefore has two steps. In the forward step, it incorporates information from past observations at every time point to compute $a^{(n)}(x_n) = p(x_1, \ldots, x_n, y_n)$. It is straightforward to show that this can be computed recursively as

$$a^{(n)}(x_n) = \pi(y_n \mid x_n) \sum_x a^{(n-1)}(x) p_{x x_n}$$

with appropriate initialisation for $a^{(1)}(x)$. The backward step considers future observations in order to obtain $b^{(n)}(x_n) = p(y_{n+1}, \ldots, y_N \mid x_n)$, which can similarly be computed

recursively. It can be shown that combining the results of the two steps gives us

$$p(x_n \mid y_1, \ldots, y_N) \propto a^{(n)}(x_n)\, b^{(n)}(x_n)$$

The computational cost of this algorithm depends on both the size of the state-space and the number of time steps. Specifically, a simple analysis shows that the complexity is $O(NM^2)$, where M is the number of states.

A simpler variant of this scheme can be used if the goal is to simply sample from $p(x_n \mid y_1, \ldots, y_N)$, that is, to obtain a path from the system conditioned on the observations. In this case, the algorithm is referred to as Forward Filtering - Backward Sampling (FFBS).

2.6 Inference for stochastic systems

Various approaches have been proposed for inference in CTMC-based systems following the ideas and principles described in the previous section. In this thesis, we propose MCMC methods developed based on the stochastic dynamics of a CTMC. This section gives an overview of different approaches suggested in the literature. Most of them include some kind of sampling scheme based on approximations to the process to facilitate inference, although other approaches are also discussed.

Hereafter, we will be primarily concerned with pCTMCs, and will use the term CTMC or MJP to refer to this particular subclass of process unless otherwise specified. Furthermore, we will only consider homogeneous CTMCs. The general problem is as follows. Assume that we have observations $D = \{(t_1, y_1), (t_2, y_2), \ldots, (t_n, y_n)\}$ of the values of a CTMC, i.e. that at each of n time points t_i, the state has been observed as y_i. The observations can be noisy measurements of the state: let $p(y_i \mid x_i)$ be the probability of measuring y_i when the state at the same time is x_i. We describe the CTMC using a set of parameters θ (which control, for example, the rates of different transitions) and we wish to compute the posterior distribution $p(\theta \mid D)$. We assume that our prior beliefs about the parameters are expressed as a distribution $p(\theta)$.

2.6.1 Direct solution

According to methodology described in the previous section, both analytical calculation of the posterior and sampling methods require computation of the likelihood. We show how this could be done in the CTMC setting based on the dynamics described earlier.

The Markovian nature of the chain allows us to decompose the likelihood into simpler terms:

$$p(D \mid \theta) = p(y_1;t_1)p(y_2;t_2 \mid y_1;t_1)\ldots p(y_n;t_n \mid y_{n-1};t_{n-1}) \tag{2.11}$$

Each of these terms is essentially a transient probability, which can in principle be obtained by solving the CME (2.2). For a finite state-space, a closed-form solution was given in (2.3). However, it involves exponentiation of a matrix, the computational cost of which increases sharply with the number of states — and as mentioned before, the number of states can grow fast. The analysis in [88] indicates cases in which the CME can be solved analytically, corresponding to very simple processes. In general, therefore, the ODE solution must be numerical. Even that, however, poses problems: the number of states can grow quickly, resulting in a large number of coupled equations — and, naturally, if the state-space is unbounded, there would theoretically be an infinite number of ODEs to be formulated and solved. An alternative method for calculating transient probabilities is given by the *uniformisation* technique [62]. This involves expressing the probability as an infinite sum and computing an approximation to it via an iterative algorithm [34]. It is still only applicable to systems with finite spaces, however. Yet another option would be to use simulation: since the SSA simulates the CTMC exactly, running it multiple times and aggregating the resulting paths in principle allows an approximation of the transient distribution at the time of interest. However, different runs can vary significantly, and so a large number of SSA executions may be required in order to compute the likelihood with reasonable accuracy, particularly when rare events can occur. Different methods must therefore be employed to avoid direct computation of the likelihood in this way.

2.6.2 Auxiliary variable samplers

To sidestep the difficulty of the likelihood computation, one can think of introducing additional variables. To see why this is useful, assume that the true path that gave rise to the observations was known. In that case, the likelihood could be computed simply by considering the noise terms. The true path is of course unknown, but it can be included in the problem using additional variables. These variables can then be sampled as part of an MCMC scheme, effectively being treated as parameters themselves.

This is the approach taken in [4], where, in addition to the parameters, a full path is sampled at every step of a M-H algorithm. This permits a calculation of the likelihood based on the auxiliary variables. Previously, Boys *et al.* had explored a similar direction

in [17]. In that case, they introduced variables representing the state of the process between observations. This addition allowed them to formulate simple conditional distributions for the variables involved and develop a Gibbs sampling scheme. Despite the simplifications it allows, the introduction of so many auxiliary variables results in a very high-dimensional search space which is hard to explore efficiently, reducing the performance of these approaches.

Rao & Teh have also proposed an algorithm in a similar direction. In [109], they showed how to construct a Gibbs sampler that efficiently samples paths from a fully-known CTMC conditioned on observations. With appropriate choice of priors on the parameters (exit rates and jump probabilities) of the chain, a parameter update step can be included in order to jointly sample from the state and parameter space [110]. A more detailed description of their algorithm is given in Chapter 4, where we develop an algorithm based on similar ideas.

2.6.3 Approximate Bayesian Computation

A different methodology for the treatment of problems with hard-to-compute likelihoods has emerged in Approximate Bayesian Computation (ABC, [125]). ABC is not a single algorithm but rather a generic approach that can be embedded into different inference schemes. Its main idea is to replace the calculation of the likelihood with simulation of the model. The simulated results are then compared to the observations, using some user-defined summary statistic and distance metric.

Here, we describe how ABC is used in a M-H algorithm and the CTMC setting. The general setup of Algorithm 2 is used, but, instead of computing the likelihood, a sample path is drawn from the model using the SSA. This is compared to the observations by, for example, taking the average Euclidean distance between the sample path and the observed values at the observation times. The resulting distance is used as a proxy for how good a fit the particular parameters are to observations: if the distance is above a threshold ε, the parameters that were used in the simulation are rejected; otherwise, they are accepted with a probability similar to the standard M-H ratio, but only considering the prior and proposal densities.

The ABC approach is, in principle, well-suited to the CTMC setting, where simulation is simple. An additional benefit is that it can easily handle infinite state-spaces, since the SSA is still applicable in those cases even if the likelihood cannot be computed. However, it suffers from its many tunable parameters: the choice of summary statistic

and distance metric can crucially affect its performance, as can the distance threshold used. In particular, as $\varepsilon \to 0$, the results obtained are exact, but acceptance becomes much more rare. In general, the samples returned come from an approximation to the true posterior. The optimal choice of threshold is not simple, as the trade-off between accuracy and performance is hard to quantify and analyse. Recent work has sought to strengthen the theoretical rigour to the ABC approach, and provide heuristics for choosing good summary statistics and threshold values — relevant reviews of the area include [56, 84].

2.6.4 Continuous approximations

Due to the complexities of the stochastic dynamics, a broad class of methods have been proposed that approximate the stochastic process with a simpler one. Often, the state of the approximating process is taken to be a continuous version of the (discrete) state-space of the original. In chemical systems, for instance, the count of molecules would be replaced by their concentration in a given volume. This continuous space allows the use of other established techniques to compute the likelihood and, consequently, perform inference. The approximation can be done in a number of ways, affecting the accuracy and the computational efficiency of the available solutions.

The simplest way is to approximate the dynamics by the mean behaviour of the stochastic process. That is, the system is characterised by a variable Y such that $Y(t) = \mathbb{E}[X(t)]$. In this deterministic approximation, the evolution of Y can be described via appropriately constructed ODEs. The resulting system is simple and much easier to treat using basic methods — based on numerical solution of ODEs — but the fact that it is deterministic makes it generally unsuitable for describing a stochastic system: apart from quantitative errors introduced by the approximation, studies have shown that such deterministic approximations can exhibit quantitatively [79] or even *qualitatively* [107] different behaviour when compared to the original stochastic systems. This is particularly the case when low counts of species are present: in the limit of large populations, X becomes more deterministic and Y matches the behaviour of a normalised version of X, with theoretical work describing necessary conditions for this to happen [74]. For low populations, however, the impact of the discrete nature of the process is very significant, and is not captured well by this kind of continuous deterministic approximation, which averages over individual behaviour. It is important, therefore, for the approximating system to incorporate (some of the) randomness of the

original in its formulation.

A first category of methods work by considering a diffusion process, described by a Stochastic Differential Equation (SDE) called the Chemical Langevin Equation. The work by Golightly & Wilkinson (2005) first detailed how to construct this process so that it corresponds to a given CTMC. The work focused specifically on the case of biochemical reaction networks, where each reaction has an associated parameter to be estimated. Stochasticity is incorporated through a white noise term in the resulting SDE. Recasting the problem in this way benefits from the extensive body of work concerning parameter inference for diffusion processes, especially applied to financial models, in both the maximum likelihood [103] and Bayesian settings [31, 29]. The solution they propose requires imposing a discretisation of time, introducing unobserved variables at the inter-observation time points in addition to the unknown parameters. This results in a high-dimensional space, making M-H sampling inconvenient. Their method contains a Metropolis-within-Gibbs step, since not all the distributions involved can be sampled from directly. The approximation introduces some error, which can be seen by the discrepancy in results when simulated compared to the original process; the authors conclude, however, that it is sufficiently accurate as far as inference is concerned.

Other methods take a different direction by employing the Linear Noise Approximation (LNA). This is rooted in a low-order Taylor expansion of the CME of the original CTMC. The result is that, by assuming higher-order terms are negligible, the single-time marginal distribution over states is approximated with a Gaussian distribution $\mathcal{N}(Y(t), V(t))$ whose mean and covariance change over time. Specifically, the mean is exactly the mean-field approximation described above. The stochasticity of the original is now captured by the covariance, which allows for the value of the process to fluctuate from this mean. Once again, thanks to the simplification afforded by the approximation, the evolution of the mean and covariance can be described in a system of closed form ODEs. Komorowski *et al.* [70] first demonstrated how the LNA can be used to compute the likelihood and thus perform inference, using a M-H sampler. At around the same time, Ruttor & Opper [113] presented a different derivation of the equations to calculate the likelihood based on a forward-backward approach. They demonstrated their results on two simple systems, employing different parameter estimation strategies. The same task was also treated by Fearnhead *et al.* in [32]. In addition to considering partially observed systems (i.e. cases where only some components of the state are observed) and presenting appropriately modified equations, they also propose a different solution scheme which yields improved accuracy. The difference lies in restarting the LNA after

each observation, instead of solving the ODEs once for the whole time interval under consideration. The authors suggest that this avoids problems stemming from degrading accuracy of ODE solutions over large horizons. Stathopoulos and Girolami [121] use the LNA as part of a more elaborate MCMC scheme, which automatically tunes the proposal steps to improve convergence.

2.6.5 Moment-based methods

Given the difficulty of solving the CME to compute the distribution over states at any time, an alternative approach is to analyse its moments. In a way, this method lies between the direct solution and the continuous approximations discussed above: the state is assumed to be discrete and the full stochastic dynamics are used, but the outcome is some simplified description of the distribution rather than the full distribution itself.

The n-th order central moment of a random variable X is defined as:

$$\mu_n = \mathbb{E}[(X - \mathbb{E}[X])^n] \tag{2.12}$$

As a quantity associated with the distribution, the value of each moment will change over time, its evolution following some ODE. The ODE for the mean is similar to that for the deterministic equivalent discussed in the previous section. More information can be obtained by considering higher-order moments, such as the covariance and skewness. Engblom [30] has shown how to extract the corresponding ODEs from the CME. However, a problem arises when trying to compute moments when the rate functions are non-linear: in this case, the ODEs for the moments of order n will depend on moments of order $n+1$. Consequently, it is generally not possible to formulate a finite set of equations, and one must therefore make use of so-called *moment closure* techniques. These eliminate the dependence on higher-order terms, resulting in a system that can be solved numerically. Different closure techniques have been proposed and used, each with their own underlying assumptions, not always accurate — their use therefore introduces some error in the results. Recent work [115, 116] has examined the validity of various such approximations when applied to the analysis of CTMCs.

These techniques, and particularly their use for parameter inference, has been explored mostly with regard to specific models. Krishnarajaha *et al.* [73] proposed several second- and third-order moment closures, applied to different epidemiological models, and showed how they can be used to obtain maximum likelihood estimates and confidence intervals for parameters. In the Bayesian setting, Gillespie & Golightly [45] used a Gaussian moment closure applied to an ecological model of aphid populations. This

permits approximate calculation of the likelihood and inference via a M-H algorithm. In a similar spirit, Capistrán *et al.* [19] examined a simple epidemiological model, deriving the first two moments and matching them to a special-purpose distribution. A more general treatment in the context of biochemical reaction networks is given in [89]: a Gaussian closure is employed to compute the moments, and the result is used to approximate the likelihood as part of a simple M-H scheme. That work includes some novelty in its treatment of processes with a large number of unobserved species, describing an efficient way of proposing values for the missing data. The method showed good results when evaluated on synthetic data for simple systems, and comparable performance with inference schemes based on a diffusion approximation. Finally, a new scheme was recently proposed [13] which chooses between different moment closures and adapts the choice according to the parameter value examined at that time, albeit for maximum likelihood estimates.

2.6.6 Particle methods

Sequential Monte Carlo or Particle Filtering is a technique for inferring the hidden state of dynamical systems. Like other Monte Carlo methods, the idea is to use a set of samples, called particles, to approximate a distribution; the key difference is that this set of particles is evolved over a number of steps until, finally, they are an approximation to the distribution of interest. More specifically, the aim is to approximate the distribution $p(x_1,\ldots,x_n \mid y_1,\ldots,y_n)$ where x_i is the state at t_i, the time of observation y_i. We start from a set of particles which represent the initial distribution $p(x_0)$. At each step, we have a particle representation $\mathcal{X}$ of $p(x_1,\ldots,x_i \mid y_1,\ldots,y_i)$. For each successive observation, this will be updated to $p(x_1,\ldots,x_{i+1} \mid y_1,\ldots,y_{i+1})$ using the following steps:

1. Choose a particle x from $\mathcal{X}$.

2. Evolve it until the next time t_{i+1} via the system dynamics (for example, by running the SSA).

3. Keep the last state (corresponding to time t_{i+1}) and add it to x. This gives a new particle x' from the distribution $p(x_1,\ldots,x_{i+1} \mid y_1,\ldots,y_i)$.

4. Calculate a weight w for the particle according to the measurement model $p(y_{i+1} \mid x_{i+1})$.

5. Repeat steps 1–4 to get a set of N weighted particles.

6. Resample from that set according to the weights. This gives a set of (uniformly weighted) particles from $p(x_1, \ldots, x_{i+1} \mid y_1, \ldots, y_{i+1})$.

One of the main shortcomings of this bootstrap particle filter algorithm [54] is the so-called particle degeneration problem, i.e. the tendency of particles to become very similar as the number of time steps grows. Solutions and adaptations have been proposed to address this, and the algorithm has become a standard tool for sampling from the posterior process $x_{1:n} \mid y_{1:n}$ of a dynamical system. A major drawback is that its accuracy can depend heavily on the number of particles N used in the approximation: a large number may be required to achieve good accuracy, but this impacts the computational performance of the algorithm.

The particle filter can be used as part of a parameter inference scheme, as shown by Andrieu *et al.* in the definition of the Particle Marginal Metropolis-Hastings (PMMH) algorithm [7]. The idea is to sample from the joint space of parameter and state values using a M-H algorithm. This scheme uses a particular proposal so that the acceptance ratio does not involve the latent state values, making it easier to compute. A particle filter is run at each step to obtain unbiased estimates of the other quantities involved. The resulting pseudo-marginal algorithm samples from the correct distribution of parameters (and state values at the observation times), as described in Section 2.5.2.2.

Owen *et al.* [101] consider combining this approach with ABC: they propose to run an ABC sampling scheme to obtain an approximation of the posterior distribution, which is then used to initialise a number of Particle MCMC chains, as described above. The idea is that the chains can be run in parallel and will converge quickly because of the informative initialisation, improving computational performance. Hajiaghayi *et al.* [57] propose an efficient way of performing the calculations in the case of CTMCs, focusing mainly on the particle filtering step but also describing how it can be used for parameter inference. Another application appears in [133], where Zechner *et al.* consider the problem of inferring the parameters governing cellular processes from measurements of a whole population of cells, where some of the parameters are unique to each cell and others are shared between the population. They include a particle filter as part of a more elaborate inference scheme that respects the shared or individual nature of each parameter.

Particle MCMC methods have also been used in conjunction with the continuous approximations discussed previously. The PMMH algorithm employed in [49] simply

uses the dynamics of the diffusion approximation instead of the discrete stochastic system when sampling a state path. In [48], both the diffusion approximation and the LNA are considered as a way to improve the performance of a two-stage scheme. First, the likelihood for a proposed parameter value is computed under one of the two continuous approximations. Then, if this is high enough, the true likelihood is estimated under the true stochastic dynamics using a standard particle filter, and this is used in the usual M-H acceptance probability. This "delayed-acceptance" scheme has the goal of weeding out unpromising parameter proposals before the expensive task of computing the likelihood.

2.6.7 Other approaches

A technique which has not been discussed in the above is variational inference. The idea behind it is to choose from a family of distributions the one that best approximates the distribution of interest (in this case, the posterior). In contrast with sampling methods like MCMC, it returns an analytical expression for the posterior, but an approximate one. Although sampling methods have received much more attention in this setting, Opper & Sanguinetti [100] have shown how to construct a variational approximation for CTMCs.

Another interesting approach taken in [22] follows a very different direction. The analysis shows how properties of CTMCs can be exploited to infer structural information about the system in question. In the biochemical context, this amounts to recovering the structure of the interactions, i.e the species involved in each reaction and the stoichiometry matrix. The key idea is that, for a given starting state, the set of possible states after an unknown number of reactions lie in a space that can be characterised geometrically. Based on observations at intermittent times, then, the possible jump vectors of individual reactions can be identified. Although primarily focused on reconstructing the structure of the reactions, this method also permits parameter learning, under the assumption that reaction rates follow the law of mass-action.

Bortolussi and Sanguinetti [16] have examined the problem of learning parameters of stochastic systems when the observations are available not as measurements but as binary data, indicating the satisfaction or not of logical properties over multiple runs of the system. Their approach is to consider the satisfaction probability of a formula (expressed in a temporal logic) as a function of the parameters. They then learn this dependence using functional approximation techniques from machine learning. This allows them to evaluate the likelihood and perform inference.

Chapter 3

The Probabilistic Programming Process Algebra

This chapter introduces the Probabilistic Programming Process Algebra (ProPPA), a language for the description of continuous time, discrete state stochastic systems with uncertain parameters. It incorporates ideas from probabilistic programming (namely, the description of uncertainty and specification of observations) in the formal framework of a process algebra. The presentation starts by describing the syntactic elements that make up a ProPPA model. The semantics of the language is then defined in terms of transition relations and a labelled transition system, followed by a discussion on appropriate underlying objects (generalisations of CTMCs) that can effectively capture the uncertainty in the model. The type of observations that we consider and the role of inference are given particular attention. A preliminary version of this chapter appeared in [40].

3.1 Background

This section introduces the FuTS (state-to-function transition systems) framework [26], which is used in the remainder of this chapter. This formalism was developed to describe various types of systems, including deterministic, non-deterministic and probabilistic, using a common framework.

In a FuTS, the possible transitions from a state s are represented collectively as $s \overset{\alpha}{\rightarrowtail} f$, where f is called the *continuation* and is a function over states. The value of $f(s')$ then captures some information about the transition $s \xrightarrow{\alpha} s'$. Depending on the codomain of the continuation functions, FuTS can represent different kinds of behaviour

and their associated information, such as non-determinism (continuations take boolean values to denote possible next states), discrete time Markov systems (values in $[0,1]$ give transition probabilities) or continuous time Markov systems (values in $\mathbb{R}$ to denote transition rates). The notation $[s_1 \mapsto v_1, s_2 \mapsto v_2, \ldots, s_n \mapsto v_n]$ is shorthand for a function f such that $f(s_i) = v_i$, $i = 1 \ldots n$ and $f(s)$ takes the zero value of its codomain (0 for real values, *false* for boolean etc.) for all states besides the specified s_i.

Essentially, FuTS are a different way of expressing operational semantics, and can capture the same information as a small-step style description. As ProPPA represents uncertainty using probability distributions, this alternative style, which already makes use of functions, seems a natural fit for expressing its semantics.

3.2 ProPPA systems

ProPPA follows an approach similar to Bio-PEPA, where interacting systems are described in terms of biochemical reactions. Although the terminology and viewpoint is biologically inspired, this does not limit the possible application fields. The main components on which a Bio-PEPA model is built are *species* and *reactions*, and ProPPA inherits the same modelling style. Intuitively, species represent the different kinds of agents in the system, while reactions describe the ways in which agents can interact. The rates at which reactions occur are governed by parameters, which can be known or uncertain. We also consider *observations* of the behaviour of the system, which can be used to perform *inference*.

Example. To illustrate the ideas described in this chapter, we will use as a running example a model of the spreading of rumours [16], a variant of the commonly-used SIR epidemiological model [69]. The agents involved, through whose interactions the rumours are spread, can be in one of three states: Ignorant, Spreader and Repressor. Ignorant individuals are those unaware of the rumour, Spreaders are actively spreading it through contact with Ignorants, while Repressors stop the spreading. We assume that only new rumours are deemed worthy of spreading; thus, when two Spreaders meet each other, they realise they are both spreading the same rumour and one of them stops, becoming a Repressor. The spreading can also stop by contact of a Repressor with a Spreader. We also assume that the number of individuals is constant, reflecting a fixed population, and that there are no new rumours — therefore, agents cannot revert to being Ignorant or Spreaders. The diagram in Figure 3.1 summarises these interactions.

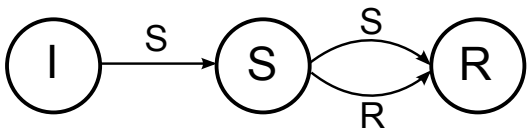


Figure 3.1: State transitions of a rumour-spreading agent. The arrow labels indicate the type of agent that must be encountered for the corresponding transition to take place.

In ProPPA, this will be modelled using three distinct species, named I, S and R. Each kind of interaction is modelled as a reaction: *spread* represents the contact of an Ignorant with a Spreader, *stop1* is the meeting of two Spreaders and *stop2* is the meeting between a Spreader and a Repressor. □

As will be shown below, the components of a model are built by combining different species. At the same time, each such language term can also be viewed as a numerical vector containing the counts of each species (assuming some arbitrary but well-defined ordering of the species, which is not otherwise important). In the following, we use the term *state* to denote either viewpoint, with the meaning being clear from the context. When we want to explicitly map a language term P to the equivalent numerical vector, we will write $\mathbf{vec}(P)$.

3.3 Syntax

A ProPPA model has five parts, which must come in a prescribed order, as described below. Illustrating these, Figure 3.2 contains the ProPPA description of the example system discussed above.

3.3.1 Parameter definitions

Parameters affect the dynamics of the system, and can be one of two types. *Concrete* parameters have a fixed numerical value, which must be given when they are declared. They are used simply for convenience, such as to facilitate experimenting with different values when writing a model. *Uncertain* parameters, on the other hand, represent unknown quantities. Instead of a value, their declaration includes a probability distribution describing the prior beliefs held about the parameter. This *prior distribution* can originate, for example, from domain knowledge or imposed constraints. It is important to note that this uncertainty does not mean that the value of the parameter changes throughout the execution of a model, but rather that it is not *a priori* known. Uncertain

```
1  k_s = Uniform(0,1);
2  k_r = Uniform(0,1);
3
4  kineticLawOf spread : k_s * I * S;
5  kineticLawOf stop1 : k_r * S * S;
6  kineticLawOf stop2 : k_r * S * R;
7
8  I = (spread,1) ↓ ;
9  S = (spread,1) ↑ + (stop1,1) ↓ + (stop2,1) ↓ ;
10 R = (stop1,1) ↑ + (stop2,1) ↑ ;
11
12 I[10] ⋈* S[5] ⋈* R[0]
13
14 observe('trace');
15 infer(ABC);
```

Figure 3.2: ProPPA model of the rumour spreading example, showing the different parts of a model: parameter definitions (lines 1-2), reaction definitions (l. 4-6), species definitions (l. 8-10), initial state (l. 12), observations and inference (l. 14-15)

parameters are used in the definition of kinetic laws, as detailed below, whereas concrete parameters can be used whenever a numerical value is needed. In particular, this means that the initial populations cannot refer to uncertain parameters.

The relevant definitions have the form

$$\begin{aligned} \mathit{parameter\_def} ::= \ & \mathit{num\_value} \\ & \mid \mathit{distr\_def} \end{aligned}$$

where $\mathit{num\_value} \in \mathbb{R}$ is used to define concrete parameters and $\mathit{distr\_def}$ is a probability distribution with its associated parameters, e.g. $\mathit{Gaussian}(0,1)$ or $\mathit{Exponential}(10)$.

3.3.2 Reaction definitions

Every reaction has a name and a kinetic law, which is a function $f : \mathbb{N}^M \times \mathbb{R}^K \to [0,\infty)$, where M is the number of species and K is the number of parameters. The kinetic law controls the rate at which the reaction happens, according to the usual stochastic dynamics: if the state of the system is $\boldsymbol{s}$ and the parameters have values $\boldsymbol{k}$, the waiting time before a reaction with kinetic law f happens is exponentially distributed with mean $\frac{1}{f(\boldsymbol{s},\boldsymbol{k})}$ (equivalently, the reaction occurs on average every $f(\boldsymbol{s},\boldsymbol{k})$ time units).

A reaction definition has the form

$$\mathit{react\_def} ::= \texttt{kineticLawOf}\ \mathit{react} : \mathit{function\_def}$$

where *react* is a reaction name and *function_def* is a function definition.

3.3.3 Species definitions

A species definition has the form

$$\begin{aligned} \mathit{species\_def} ::= {} & \mathit{atomic\_def} \\ & \mid \mathit{atomic\_def} + \mathit{species\_def} \\ \mathit{atomic\_def} ::= {} & (\mathit{react}, n)\mathit{op} \\ \mathit{op} ::= {} & \uparrow \mid \downarrow \mid \oplus \mid \ominus \mid \odot \end{aligned}$$

where *react* is a reaction name and $n \in \mathbb{N}^*$.

A species is defined by listing its possible behaviours. An atomic behaviour describes the participation of a species in a single reaction, distinguishing what its role is. There are five possible roles, inherited by the equivalent Bio-PEPA ones. A *reactant*, indicated by the operator $\downarrow$, is a species whose quantity is reduced when a reaction occurs. The $\uparrow$ operator indicates a *product* species, i.e. one whose amount increases. The remaining three roles concern *modifiers*, species which are involved in a reaction and thus affect its rate, but whose amount remains unchanged by the reaction. These roles can arise, for instance, when modelling biochemical processes involving catalysts and inhibitors of reactions, but can also represent more general situations, such as aspects of the surrounding environment. The corresponding symbols are $\oplus$ and $\ominus$, according to whether the species affects the rate in a positive or negative way, respectively, and the more general $\odot$ which does not specify the exact effect of a species.

The definition additionally contains a *stoichiometry n*, the net quantity by which the count of the species is affected by the reaction. The intuition and terminology are easier to understand in a biological setting. For a reactant, this is the amount of the species that is consumed by the reaction. For a product, it is the amount produced. For a modifier, it is the minimum amount required for that species to take part in the reaction.

The choice operator $+$ allows the combination of alternative atomic behaviours, for species which are involved in multiple reactions.

3.3.4 Initial state

This corresponds to the *model component* of PEPA and Bio-PEPA, and describes the initial configuration of the system. We have chosen to keep the same syntax, making use of the *cooperation* operator $\underset{\mathcal{L}}{\bowtie}$, where $\mathcal{L}$ is a set of reactions. The meaning of the operator, informally, is that components joined in this way perform the specified reactions jointly. In all the examples considered here, we use the notation $\underset{*}{\bowtie}$, indicating that the joined components cooperate on all the reactions they share with each other. The initial state is specified as follows:

$$\begin{aligned} \mathit{init\_state} ::= \ & \mathit{comp}[n] \\ & \mid \mathit{comp}[n] \underset{\mathcal{L}}{\bowtie} \mathit{init\_state} \end{aligned}$$

where $n \in \mathbb{N}$ and $\mathcal{L}$ is a set of reaction names. The notation $S[n]$ denotes n copies of species S or, viewed differently, that the level of S is n.

3.3.5 Observations, inference and configuration

This part of the model specifies the observations of the system, and which inference method should be used. The parameters of the chosen algorithm can optionally be tuned. The relevant keywords (`observe`, `infer` and `configure`) are discussed in more detail in Section 3.5.

We can now give the formal definition of a ProPPA model:

Definition 6. A ProPPA model $\mathcal{M}$ is a tuple $\langle Comp, \mathcal{K}_c, \mathcal{K}_u, \mathcal{F}, O, P\rangle$, where:

- $Comp$ is the set of species definitions;
- $\mathcal{K}_c$ is the set of concrete parameters;
- $\mathcal{K}_u$ is the set of uncertain parameters;
- $\mathcal{F}$ is the set of kinetic laws;
- O are the observations; and
- $\mathcal{P}$ is the model component.

Following [35], we call $\mathcal{T} = \langle Comp, \mathcal{K}_c, \mathcal{K}_u, \mathcal{F}, O\rangle$ the *context*, and will often write a model as $\mathcal{M} = \langle \mathcal{T}, P\rangle$.

3.4 Semantics

To define the semantics of a ProPPA model, we start by constructing two transition relations and their corresponding LTS. This approach follows that of Bio-PEPA, which uses a two-step semantics; indeed, we keep the name of the two transition relations defined for Bio-PEPA.

3.4.1 The capability relation

The first of them is the *capability relation* $\rightarrow_c$, which concerns the possible behaviour of a (simple or compound) component. Informally, $P \xrightarrow{l}_c S$ means that a component P can become S; the action that corresponds to this transition, along with some other information, is captured in the label l. The capability relation is qualitative: it expresses only whether a transition is possible and does not give any information about its rate. It is therefore, in a way, of "structural" concern, a consequence purely of the species definitions.

Since ProPPA models only include uncertainty in the kinetic parameters, not in the structure of the interactions (i.e. not in the species definitions), it follows that the capability relation does not include any such uncertainty either; any non-determinism found in it is a consequence of the choice and cooperation operators. For this reason, the capability relation is identical to the one for Bio-PEPA, originally defined in [23] in the standard structured operational semantics style. For the sake of coherence with the next subsections, we reformulate it in terms of the FuTS framework, presented in Figure 3.3. The capability relation is the smallest relation that satisfies these rules.

The first three rules, `PrefixReac`, `PrefixProd` and `PrefixMod`, define the behaviour of a simple component that is a reactant, product or modifier, respectively, in a single reaction. Such a component can only perform one action, as indicated by the "singleton" notation $[S \mapsto \mathit{true}]$ in these rules. The transition label contains two elements: the reaction name, and a list of roles. Each element of this list has the form $S : op(l,k)$, with $op \in \{\downarrow, \uparrow, \oplus, \ominus, \odot\}$ and $l,k \in \mathbb{N}$. This indicates that species S has role op and an initial level of l, which will change by k.

The rules `Choice1` and `Choice2` formalise the meaning of the choice operator $+$. As mentioned above, a component $P_1 + P_2$ can perform all actions that either P_1 or P_2 can perform.

The semantics of the cooperation operator is defined in the rules `Coop1`, `Coop2` and `Coop3`. For the behaviour of $P_1 \underset{L}{\bowtie} P_2$ when considering an action α, we make a

$$\texttt{PrefixReac} \quad (a,k) \downarrow S(l) \quad \overset{(a,[S:\downarrow(l,k)])}{\succ\!\!\longrightarrow_c} \quad [S(l-k) \mapsto \mathit{true}] \quad l \geq k$$

$$\texttt{PrefixProd} \quad (a,k) \uparrow S(l) \quad \overset{(a,[S:\uparrow(l,k)])}{\succ\!\!\longrightarrow_c} \quad [S(l+k) \mapsto \mathit{true}] \quad l \geq 0$$

$$\texttt{PrefixMod} \quad (a,k)\ op\ S(l) \quad \overset{(a,[S:op(l,0)])}{\succ\!\!\longrightarrow_c} \quad [S(l) \mapsto \mathit{true}] \quad l \geq k$$

$$\texttt{Choice1}\ \frac{P_1 \overset{(a,w)}{\succ\!\!\longrightarrow_c} f}{P_1 + P_2 \overset{(a,w)}{\succ\!\!\longrightarrow_c} f} \qquad \texttt{Choice2}\ \frac{P_2 \overset{(a,w)}{\succ\!\!\longrightarrow_c} f}{P_1 + P_2 \overset{(a,w)}{\succ\!\!\longrightarrow_c} f}$$

$$\texttt{Coop1}\ \frac{P_1 \overset{(a,w)}{\succ\!\!\longrightarrow_c} f_1 \qquad a \notin \mathcal{L}}{P_1 \underset{\mathcal{L}}{\bowtie} P_2 \overset{(a,w)}{\succ\!\!\longrightarrow_c} g_1} \qquad \texttt{Coop2}\ \frac{P_2 \overset{(a,w)}{\succ\!\!\longrightarrow_c} f_2 \qquad a \notin \mathcal{L}}{P_1 \underset{\mathcal{L}}{\bowtie} P_2 \overset{(a,w)}{\succ\!\!\longrightarrow_c} g_2}$$

$$\texttt{Coop3}\ \frac{P_1 \overset{(a,w_1)}{\succ\!\!\longrightarrow_c} f_1 \qquad P_2 \overset{(a,w_2)}{\succ\!\!\longrightarrow_c} f_2 \qquad a \in \mathcal{L}}{P_1 \underset{\mathcal{L}}{\bowtie} P_2 \overset{(a,w_1::w_2)}{\succ\!\!\longrightarrow_c} g}$$

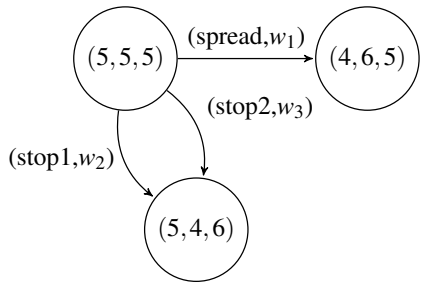
where:

$$g_1(Q) = \begin{cases} f_1(Q_1) & \text{if } Q = Q_1 \underset{\mathcal{L}}{\bowtie} P_2 \\ \mathit{false} & \text{otherwise} \end{cases} \qquad g_2(Q) = \begin{cases} f_2(Q_2) & \text{if } Q = P_1 \underset{\mathcal{L}}{\bowtie} Q_2 \\ \mathit{false} & \text{otherwise} \end{cases}$$

$$g_1(Q) = \begin{cases} f_1(Q_1) \wedge f_2(Q_2) & \text{if } Q = Q_1 \underset{\mathcal{L}}{\bowtie} Q_2 \\ \mathit{false} & \text{otherwise} \end{cases}$$

$$\texttt{Constant}\ \frac{P \overset{(a,w)}{\succ\!\!\longrightarrow_c} f \qquad Q \overset{def}{=} P}{Q \overset{(a,w[P \to Q])}{\succ\!\!\longrightarrow_c} f}$$

Figure 3.3: Capability relation semantic rules for ProPPA models expressed in the FuTS formalism.



$$w_1 = [I\downarrow\colon (5,1), S\uparrow\colon (5,1)]$$
$$w_2 = [S\downarrow\colon (5,1), R\uparrow\colon (5,1)]$$
$$w_3 = [S\downarrow\colon (5,1), R\uparrow\colon (5,1)]$$

Figure 3.4: Part of the capability relation for the example in Section 3.4.1, where a term $I[x] \underset{*}{\bowtie} S[y] \underset{*}{\bowtie} R[z]$ is shown as (x,y,z).

distinction according to whether or not α is shared between the cooperating components. If it is not, but one of them can perform α, then the other component is ignored and the $P_1 \underset{\alpha}{\bowtie} P_2$ behaves as P_1 or P_2 (rules `Coop1`, `Coop2`). If α is shared, however, both components participate in the action, and we must record the changes for both of them. This means concatenating the labels from both of their individual transitions, indicated by the :: notation (`Coop3`).

Finally, a rule `Constant` is provided for completeness and to cover the species definitions given in the model. Its intent is that, if a definition $Q = P$ is present, then Q behaves exactly as P. Because the list of roles derived for P makes reference to the component name, this must be replaced by the name of Q; we write $w[P \to Q]$ for a copy of w where every instance of P has been renamed to Q.

Example. Consider the component $I[5] \underset{*}{\bowtie} S[5] \underset{*}{\bowtie} R[5]$ in the rumour-spreading example. From this particular state, all three reactions can occur. The relevant parts of the capability relation, depicted in Figure 3.4 along with the corresponding labels, are:

$$I[5] \underset{*}{\bowtie} S[5] \underset{*}{\bowtie} R[5] \xrightarrow{(spread,w_1)}_c \left[I[4] \underset{*}{\bowtie} S[6] \underset{*}{\bowtie} R[5] \mapsto true\right]$$
$$I[5] \underset{*}{\bowtie} S[5] \underset{*}{\bowtie} R[5] \xrightarrow{(stop1,w_2)}_c \left[I[5] \underset{*}{\bowtie} S[4] \underset{*}{\bowtie} R[6] \mapsto true\right]$$
$$I[5] \underset{*}{\bowtie} S[5] \underset{*}{\bowtie} R[5] \xrightarrow{(stop2,w_3)}_c \left[I[5] \underset{*}{\bowtie} S[4] \underset{*}{\bowtie} R[6] \mapsto true\right]$$

3.4.2 The stochastic relation

If the capability relation is a qualitative description of which states are reachable from a given state, the *stochastic relation* $\to_s$ quantifies the rate at which these transitions occur. Since the parameters controlling the rates can be uncertain, the rate of a transition

is not fixed either, except in the extreme case where all the relevant parameters are concrete. Every transition will therefore be assigned a distribution over possible rates, expressing this uncertainty.

While the reachability information captured by the capability relation could be derived from just the species definitions, deriving the rates requires additional elements of the model — namely, the kinetic laws and the parameter definitions. The stochastic relation is thus defined over entire models rather than components. It builds on the capability relation, using the information contained therein; specifically, the state, which is contained in the capability relation labels, is passed on to the appropriate kinetic law. Informally, $\mathcal{M} \xrightarrow{(\alpha,\mu)}_s \mathcal{M}'$ means that $\mathcal{M}$ can transition to $\mathcal{M}'$ via reaction α, and the rate of that transition is distributed according to μ. Using the FuTS notation, this is written as $\mathcal{M} \overset{\alpha}{\rightarrowtail}_s f$ with $f(\mathcal{M}') = \mu$. A fixed rate r is expressed by the Dirac distribution $\delta(r)$, where all the probability mass is assigned to r. The degenerate case $\delta(0)$ indicates that the transition occurs at rate 0, i.e. never; this is used for completeness, when we need to describe transitions that are not allowed by the capability relation.

The stochastic relation is the smallest relation for which the following rule holds:

$$\frac{P \overset{(a,w)}{\rightarrowtail}_c g}{\langle \mathcal{T}, P \rangle \overset{a}{\rightarrowtail}_s h_{g,w,\mathcal{T}}}$$

where the function h maps models to distributions of rates:

$$h_{g,w,\mathcal{T}}(\langle \mathcal{T}', s' \rangle) = \begin{cases} \delta(0) & \text{if } \mathcal{T}' \neq \mathcal{T} \\ \delta(0) & \text{if } g(s') = \mathit{false} \\ \mu & \text{otherwise} \end{cases}$$

Essentially, h makes sure that the capability relation is respected by assigning zero rates to unreachable states or different contexts.

We now show how to derive the distribution μ over rates in the non-trivial cases. Firstly, recall that the kinetic law of α is $\mathcal{F}_\alpha(\mathbf{s}, \theta)$, generally a function of both the state and the parameters. When considering transitions of a component P, the state is fixed: the rate of leaving P via α is $T(\theta) = \mathcal{F}_\alpha(\mathbf{vec}(P), \theta)$, which depends only on the parameters θ (or a subset of them). Since θ are random variables, this rate also follows a distribution, whose pdf can be obtained by use of the following:

Lemma 1. *If X is a continuous random variable with pdf f_X and $Y = T(X)$, where T is strictly monotonic, then the pdf of Y is*

$$f_Y(y) = \left| \frac{\mathrm{d}T^{-1}(Y)}{\mathrm{d}Y} \right|_{Y=y} f_X(T^{-1}(y)) \tag{3.1}$$

To see why this is valid, let us first consider the case where T is strictly increasing, which implies that the inverse function T^{-1} is well-defined and is also increasing. The cumulative distribution function of Y is then:

$$F_Y(y) = P(Y \leq y) = P(T(X) \leq y) = P\left(X \leq T^{-1}(y)\right) = F_X\left(T^{-1}(y)\right)$$

and the corresponding pdf is:

$$f_Y(y) = \left.\frac{\mathrm{d}F_Y(y)}{\mathrm{d}Y}\right|_{Y=y} = \left.\frac{\mathrm{d}F_X\left(T^{-1}(Y)\right)}{\mathrm{d}Y}\right|_{Y=y} = f_X\left(T^{-1}(y)\right)\left.\frac{\mathrm{d}T^{-1}(Y)}{\mathrm{d}Y}\right|_{Y=y}$$

Considering the case where T is strictly decreasing is similar and leads to the result (3.1). Generalising this to multi-dimensional random variables is straightforward.

Example. Continuing with the possible reactions from the state $I[5] \underset{*}{\bowtie} S[5] \underset{*}{\bowtie} R[5]$, we denote with $\mathcal{T}$ the context defined by the model in Figure 3.2. In particular, the rates for leaving that state are:

$$\begin{aligned} r_1 &= f_{spread}(k_s) = 25k_s \\ r_2 &= f_{stop1}(k_r) = 25k_r \\ r_3 &= f_{stop2}(k_r) = 25k_r \end{aligned}$$

Knowing that $k_s \sim \text{Uniform}(0,1)$ and writing $\mu(k_s)$ for its pdf, we can compute the pdf μ_1 of r_1. The inverse transform is given by $k_s = f^{-1}_{spread}(r_1) = \frac{r_1}{25}$, with $\frac{\mathrm{d}k_s}{\mathrm{d}r_1} = \frac{1}{25}$. From Equation (3.1),

$$\mu_1(r_1) = \left|\frac{1}{25}\right| \mu\left(\frac{r_1}{25}\right) = \begin{cases} \frac{1}{25} & \text{if } 0 \leq \frac{r_1}{25} \leq 1 \Leftrightarrow 0 \leq r_1 \leq 25 \\ 0 & \text{otherwise} \end{cases}$$

This gives us the intuitive result that r_1 is distributed uniformly between 0 and 25. We can similarly obtain the same distribution for r_2 and r_3. The relevant part of the stochastic relation is then:

$$\begin{aligned} \langle \mathcal{T}, I[5] \underset{*}{\bowtie} S[5] \underset{*}{\bowtie} R[5] \rangle &\overset{spread}{\rightarrowtail}_s \left[\langle \mathcal{T}, I[4] \underset{*}{\bowtie} S[6] \underset{*}{\bowtie} R[5] \rangle \mapsto \mu_1 \right] \\ \langle \mathcal{T}, I[5] \underset{*}{\bowtie} S[5] \underset{*}{\bowtie} R[5] \rangle &\overset{stop1}{\rightarrowtail}_s \left[\langle \mathcal{T}, I[5] \underset{*}{\bowtie} S[4] \underset{*}{\bowtie} R[6] \rangle \mapsto \mu_2 \right] \\ \langle \mathcal{T}, I[5] \underset{*}{\bowtie} S[5] \underset{*}{\bowtie} R[5] \rangle &\overset{stop2}{\rightarrowtail}_s \left[\langle \mathcal{T}, I[5] \underset{*}{\bowtie} S[4] \underset{*}{\bowtie} R[6] \rangle \mapsto \mu_3 \right] \end{aligned}$$

where μ_1, μ_2, μ_3 all represent the pdf of the uniform distribution on $[0,25]$.

Based on the stochastic relation, we can refer to the set of reachable configurations of the model. This will be useful in the next section.

Definition 7. The *derivative set* $\boldsymbol{ds}(\mathcal{M})$ of a model $\mathcal{M}$ is the smallest set such that:

i. $\mathcal{M} \in \boldsymbol{ds}(\mathcal{M})$, and

ii. If $\mathcal{P} \in \boldsymbol{ds}(\mathcal{M})$ and, for some α, $\mathcal{P} \xrightarrow{(\alpha,\mu)}_s \mathcal{P}'$ with $\mu \neq \delta(0)$, then $\mathcal{P}' \in \boldsymbol{ds}(\mathcal{M})$

3.4.3 Probabilistic Constraint Markov Chains

The next step is to provide an underlying model that captures the dynamics described by the stochastic relation. In languages including PEPA and Bio-PEPA, this is done by constructing a CTMC based on the transition relations defined in the semantics. In the case of ProPPA, however, it is easy to see that CTMCs are not appropriate. The reason is that transitions between the states of a CTMC occur at specific rates, whereas the ProPPA semantics defines distributions over rates, instead, reflecting the uncertainty in model parameters. We therefore need a generalisation of CTMCs which will allow for the description of such uncertainty.

The Constraint Markov Chain (CMC) framework, described in Section 2.3.1, can provide a starting point for the object we require. CMCs, defined for discrete-time systems, handle uncertainty by specifying a constraint satisfaction function, essentially allowing or disallowing particular values of the transition probabilities. There are two limitations which stop CMCs from meeting our needs, however. The first is that the definition assumes working with discrete time, whereas we are modelling continuous-time systems; we must therefore consider not transition probabilities, but rates. The second, more subtle, reason is that while a CMC defines which values are acceptable, it says nothing about their relative likelihood. In order to match the semantics, we need to go a step further: instead of just considering the set of acceptable rates, we must place a distribution over them.

This move from the boolean view to a quantified one, along with the switch to the continuous time domain, naturally leads to the following definition:

Definition 8. A Probabilistic Constraint Markov Chain (pCMC) is a tuple $\langle S, o, A, V, \phi \rangle$, where:

- S is the set of states.
- $o \in S$ is the initial state.

- A is a set of atomic propositions.
- $V\colon S \to 2^{2^A}$ gives a set of acceptable labellings for each state.
- $\phi\colon S \times S \times [0,\infty) \to [0,\infty)$ is the *constraint function*.

A few things are worth noting about this definition. Firstly, for any pair of states s,t, the function $\phi(s,t,\cdot)$ is a probability density, which is why it takes values in $\mathbb{R}_+$. Furthermore, it is properly normalised, i.e. $\int_0^\infty \phi(s,t,r)\,\mathrm{d}r = 1$ for every $s,t \in S$. Every state can be labelled with any number of atomic propositions. Following the original definition of CMCs, we permit more than one such labellings, hence why V maps each state to a set of sets of propositions. As we are not concerned with model checking of pCMCs, we will not address this point further.

In a way, we can think of pCMC as a distribution over classic CTMCs. The continuous-time equivalent of a CMC, similarly, could be viewed as a family or set of CTMCs, each of them obtained for a specific choice of rates. Between the two, pCMCs can express richer information (akin to the difference between the probabilistic and purely non-deterministic settings), which makes them more suited to capturing the uncertainty contained in ProPPA models. With this additional expressive power, it is easy to map the LTS corresponding to the stochastic relation to a pCMC.

The pCMC will have one state for each state in the derivative set. The constraint function ϕ can be obtained by the stochastic relation, with one subtle point: there may be more than one way (reaction) of transitioning between a pair of states. In this case, all possible transitions must be considered when constructing the pCMC, and the rate will be the sum of the rates over all possible transitions. The distribution of this total rate can be computed by keeping in mind the following:

Lemma 2. *If $X_i, i = 1,\ldots,N$ are continuous random variables with respective density functions f_i, then the variable $\sum_{i=1}^N X_i$ has density $f_1 \otimes f_2 \otimes \cdots \otimes f_N$, where $\otimes$ denotes convolution of functions.*

To formalise this, we introduce some new definitions. Firstly, we define the set of possible reactions between two states s and t as

$$\mathcal{A}_{s,t} = \{\alpha \mid \exists f : s \overset{\alpha}{\rightarrowtail}_s f \text{ and } f(t) \neq \delta(0)\}$$

The distribution of the *total rate* of going from s to t is then

$$\mu_{s,t} = \bigotimes_{\alpha \in \mathcal{A}_{s,t}} f^{(\alpha)}(t), \text{ where } s \overset{\alpha}{\rightarrowtail}_s f^{(\alpha)}$$

A ProPPA model $\langle \mathcal{T}, P \rangle$ can now be mapped to a pCMC $\langle S, o, A, V, \phi \rangle$, where:

- $o = P$, the model component of the model,
- $S = \boldsymbol{ds}(\langle \mathcal{T}, P \rangle)$,
- $A = \varnothing$ and $V(s) = \{\varnothing\}\ \forall s \in S$,
- $\phi(s,t,r) = \mu_{s,t}(r)$, where $\mu_{s,t}$ is the total rate distribution defined above.

Note that, since we are not concerned with model checking, we provide no propositions or labellings. Furthermore, to match the semantics of ProPPA, we consider pCMCs as Uncertain Markov Chains (Section 2.3.1). This means that, to simulate the chain, we only ever pick a single rate for a transition. This reflects the fact that parameters in a ProPPA model have a constant value, even if that value is not known.

In the extreme case where all the constraint functions are Dirac distributions, i.e. all the rates are fully known, the pCMC essentially collapses to a CTMC. This can arise, for example, when all the parameters of a ProPPA model are concrete. In that case, the semantics matches that of a Bio-PEPA model.

Example. We focus again on the transitions from the state $s = \langle \mathcal{T}, I[5] \underset{*}{\bowtie} S[5] \underset{*}{\bowtie} R[5] \rangle$. From the stochastic relation, we know that the possible next states are

$$t = \langle \mathcal{T}, I[4] \underset{*}{\bowtie} S[6] \underset{*}{\bowtie} R[5] \rangle \text{ and}$$
$$t' = \langle \mathcal{T}, I[5] \underset{*}{\bowtie} S[4] \underset{*}{\bowtie} R[6] \rangle$$

The only way of going from s to t is via reaction *spread*, the rate of which has a distribution μ_1. There are two ways to go to t': *stop1* and *stop2*, whose rates are distributed as μ_2 and μ_3, respectively. From the previous example, we know that $\mu_1 = \mu_2 = \mu_3 = \text{Uniform}(0,25)$. In the pCMC, there will be a single transition between s and t', covering both reactions. Its rate will be distributed with pdf $\mu_{23} = \mu_2 \otimes \mu_3$, which can be seen to be a triangular or Irwin-Hall distribution over $[0,50]$ (Figure 3.5). The corresponding values of the constraint function will be:

$$\phi(s,t,r) = \mu_1(r)$$
$$\phi(s,t',r) = \mu_{23}(r)$$

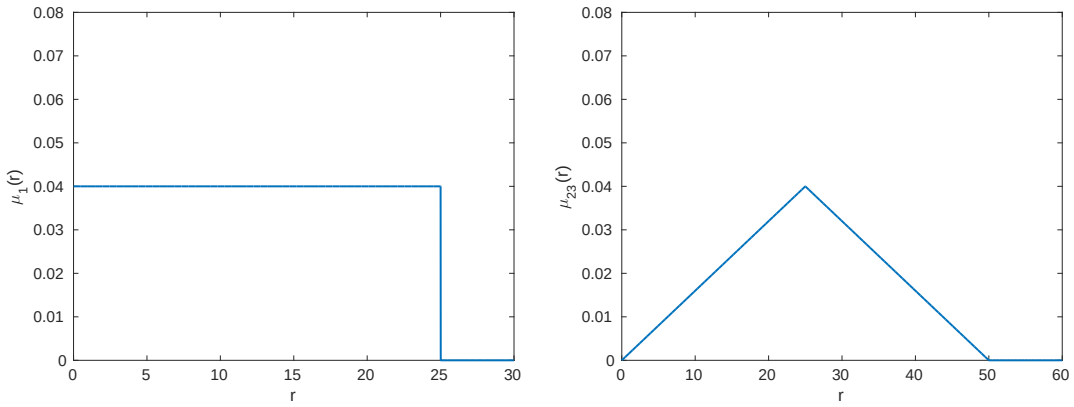


Figure 3.5: Probability density functions of the rates for the transition from s to t (left) and to t' (right) in the example model.

3.5 Observation and inference

The previous sections have discussed how uncertainty about parameters is represented and integrated into the ProPPA semantics. We will now consider the second novel contribution of ProPPA, that is, the inclusion of observations from a system, and the process of inference. Observations contain information about the behaviour of the system, which complements the knowledge specified in the priors. One can think of inference as taking advantage of this observed information, combining it with the priors, and providing a new distribution over the parameters that reflects both sources.

3.5.1 Form of observations

The `observe` keyword specifies a file containing observations from the system. The specification of the language is open-ended with respect to the form that these take; the particular details are determined by the algorithm that is to be used, as different methods act on different kinds of data. We give two examples of forms that can be used.

In many cases, the required input will be a time-series of observations from the system. This can be represented as a matrix, where the first column contains observation times, and every other column corresponds to a species; these columns contain measurements, possibly noisy, of the corresponding species at the given time. A time-series corresponds to a single experiment or run of the system, so multiple observations could be incorporated by specifying more than one file.

An alternative is to describe the observed behaviour using temporal logic formulae. Depending on the chosen logic, this could be satisfaction probabilities of certain proper-

ties, calculated over a number of experimental runs, or desired behaviour that we want to force the system to have (in order to match existing knowledge, for instance).

3.5.2 Semantics of inference

As with the form of the observations, the language definition does not specify a particular inference algorithm. This is by design, so as to have an extensible platform where different methods can be added. In general, however, inference methods fit a shared framework: their purpose is to extract information from the observations and combine it with the specification of the model, resulting in a new distribution over the parameters. More formally, an inference algorithm can be seen as a transformation of the context, changing a model $\mathcal{M} = \langle Comp, \mathcal{K}_c, \mathcal{K}_u, \mathcal{F}, O, P \rangle$ to $\mathcal{M}' = \langle Comp, \mathcal{K}_c, \mathcal{K}'_u, \mathcal{F}, \varnothing, P \rangle$.

On a practical level, the `infer` keyword declares the algorithm that is to be used. The user can control the behaviour of the algorithm by defining appropriate parameters in a configuration file, specified via the optional `configure` statement. The various algorithms and configurable parameters are detailed in Chapter 5.

3.5.3 Inference examples

We close the chapter by showing two examples of performing inference on the model of Figure 3.2, using the two different forms of input described above. We do not go into details about the algorithms used as they will be discussed in subsequent chapters. We note, however, that the first solution is a very simple approach, and that developing more efficient inference methods is the subject of the next chapter. In both cases, the prior for each of the two parameters k_r and k_s is a uniform distribution on $[0, 1]$ (Figure 3.6a).

Inference from time-series observations

We first assume that our observations are measurements of the species at different times during a run of the system. We can use an ABC algorithm that provides a simple way of exploring the parameter space and keeping those values which better fit the data. ABC is particularly useful in cases where the likelihood is unknown or intractable to calculate. The algorithm returns an approximation to the posterior distribution as a set of parameter samples.

For this experiment, we simulated the system with $k_s = 0.5$, $k_r = 0.1$, and used ten points from the resulting trajectory as the input time-series. We used the ABC-MCMC

algorithm [125] described in Section 2.6.3, tuning its parameters to achieve a reasonable acceptance rate. New model parameters were proposed from Gaussian proposals with standard deviation 0.07. For each sampled trajectory, we computed its Euclidean distance from the input; this calculation was performed for each species separately and the total distance was compared to a threshold of 6.

We gathered 100,000 parameter samples, plotted in Figure 3.6b. Considering each parameter separately yields the histograms in Figures 3.6c and 3.6d. From these, we can see that k_r is estimated better and, additionally, is distributed much more narrowly than k_s. The fact that the posterior distribution of k_s is broad is itself informative: it indicates that the observations do not allow further reduction of uncertainty or, in other words, they are consistent with a wide range of parameter values.

This can, indeed, be validated experimentally. Simulating the system for values of k_s in $[0.3, 1.0]$ shows that its behaviour does not change much in this range, supporting our results. Even with this wide variance, however, the posterior differs from the prior in that it assigns little or no probability to values of k_s in $[0, 0.3)$. In this range, the behaviour of the system becomes even qualitatively different to that in the input observations — specifically, the number of ignorants does not decrease to zero in the long term, on average. These values are therefore not favoured in the posterior.

Inference from specification

We now examine the possibility of inference from specification in the form of logical constraints, rather than quantitative observations in the form of time-series. In many cases detailed quantitative information is not readily available, whereas we have an idea of how the model should behave with respect to certain properties. We follow the framework of [16], in which observations are satisfaction values of formulae specified in a suitable temporal logic. This method uses Statistical Model Checking to obtain an estimate for the parameter value that optimises the posterior distribution. For a fuller Bayesian treatment, we can use this to approximate the whole posterior by employing the *Laplace approximation* [124]. Briefly, this involves finding a maximum of the logarithm of the posterior and calculating its second-order derivatives around that point. We then construct a Gaussian distribution that is centred on the maximum and has matching second derivatives. This Gaussian constitutes the Laplace approximation to the posterior.

For this example, we have considered the following three properties, expressed in

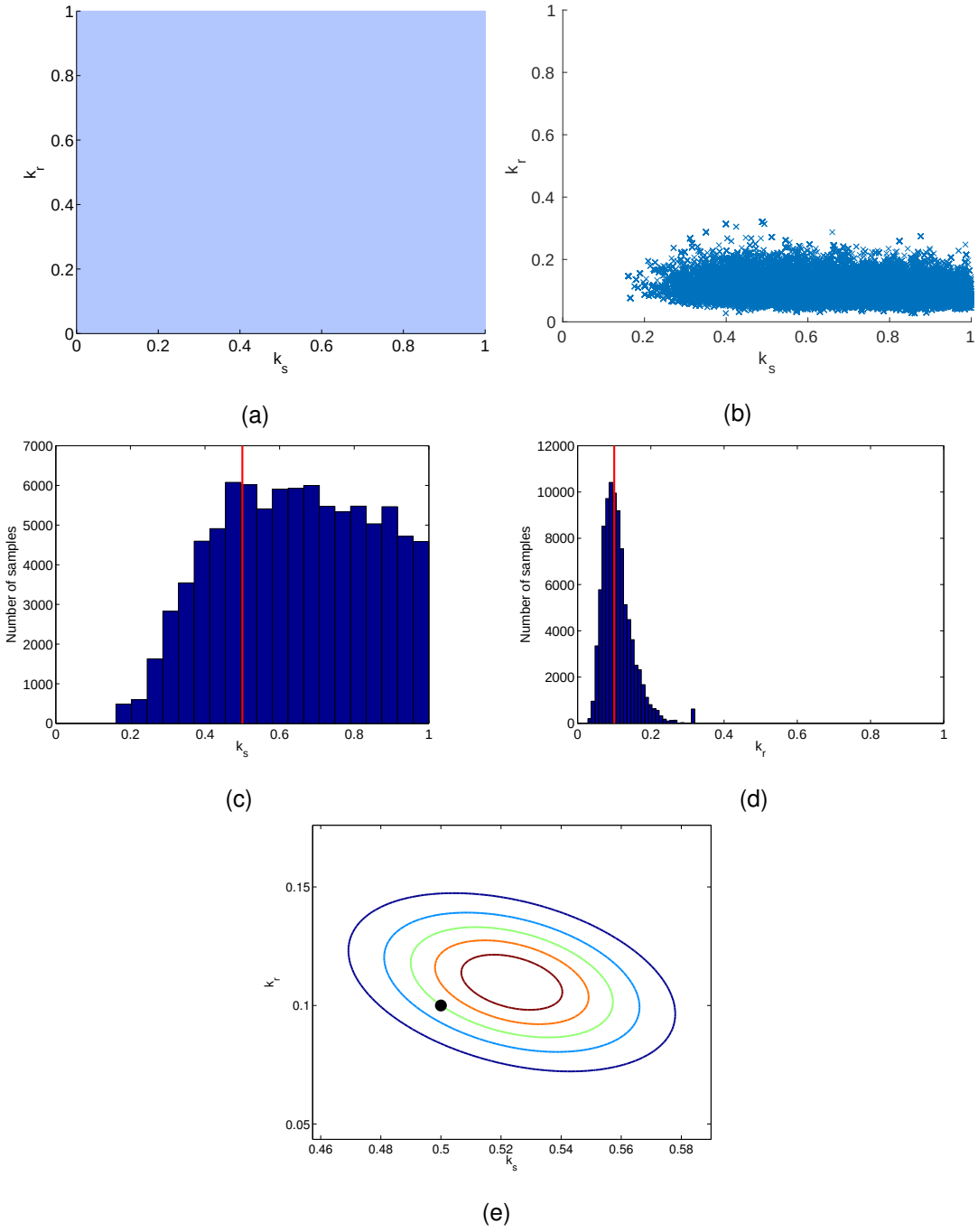


Figure 3.6: Inference results for the example model: (a) prior distributions; (b)-(d): samples from the posterior when using ABC and histograms of sample values for each parameter (red lines mark the true values); (e) contours of the Laplace Approximation of the posterior when using formulae as inputs, with the outer ring corresponding to the 98th quantile (black dot marks the true values).

Metric Interval Temporal Logic [3]:

- $\mathbf{G}^{[3,5]}(I > 0)$: There are still ignorants at all time points between time 3 and 5, i.e. the rumour has not reached everyone in the population.
- $\mathbf{G}^{[0.5,1]}(R \geq S)$: Between time 0.5 and 1, there are always more repressors than spreaders.
- $(\mathbf{F}^{[0,1]}(S \geq 7.5)) \wedge (\mathbf{G}^{[1,2]}(S \leq 3.75))$: The proportion of spreaders reaches or exceeds 50% of the population before time 1, and between time 1 and 2 the spreaders are always less than or equal to 25% of the population.

In order to produce the input specification data, we simulated the system 100 times after fixing $k_s = 0.5$, $k_r = 0.1$ as before. After running the optimisation algorithm using this input and the priors from the model, we obtained estimates of 0.5236 and 0.1098 for k_s and k_r respectively. Figure 3.6e depicts the contours of the posterior Gaussian distribution, given via Laplace approximation. Similarly to the previous section, the model appears to be less sensitive to k_s compared to k_r. This is reflected in their standard deviations, as given by the Laplace approximation; these are 0.0288 and 0.0199 respectively, meaning that the approximate distribution captures the increased uncertainty regarding k_s.

Chapter 4

Inference for MJPs via Random Truncations

This chapter explores some methods for learning MJP-based models from time-series observations, focusing on the case of processes with infinite state-spaces. We propose a new approach for dealing with such systems, building on previous work for MJP inference and combining it with a method for obtaining unbiased estimates of infinite sums. The result is a pair of algorithms that make use of pseudomarginal MCMC theory to sample from the correct posterior distribution, are more scalable than state-of-the-art methods and provide a principled treatment of open systems.

We start by formulating the problem. We consider a pMJP with state-space $\mathcal{S}$ and rates given by a function r, i.e. $r_{\boldsymbol{\theta}}(s,s')$ is the rate of jumping from s to s' for $s,s' \in \mathcal{S}$. The rates can generally depend on a finite number of parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_K)$, but we will drop the explicit dependence on $\boldsymbol{\theta}$ unless its value is not clear. We assume that each θ_j has a known prior distribution $p(\theta_j)$. We are given a set of observations $\mathcal{D} = \{(t_i, y_i)\}_{i=1}^N$, where y_i is a measurement of the process at time t_i, possibly corrupted according to some (known) noise model. Our goal is to compute the posterior distribution of each parameter, $p(\theta_j \mid \mathcal{D}), j = 1, \ldots, K$.

As we will see, the main difficulty lies in expressing or computing the likelihood. We start with the simplest, solvable case.

4.1 Metropolis-Hastings sampling for finite processes

If $\mathcal{S}$ is finite, the process can be described by its generator matrix $\mathbf{A}$ (see Section 2.1). In this case, the CME consists of a finite number of ODEs, which can be solved to give

the time-marginal of the process at any time t as:

$$\mathbf{p}_t = \mathbf{p}_0 e^{\mathbf{A}t} \tag{4.1}$$

Assume that the probability of measuring y when the state at that time is s is given by $\pi(y \mid s)$. The probability of a single observation (t, y) is then

$$p(t, y) = \sum_{s \in \mathcal{S}} p_t(s)\pi(y \mid s)$$

and the likelihood can be written as

$$p(\mathcal{D} \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} \sum_{s \in \mathcal{S}} p_{t_i}(s \mid \boldsymbol{\theta})\pi(y_i \mid s) \tag{4.2}$$

which can be computed via (4.1). This calculation of the likelihood can be embedded in a M-H scheme and used to sample from $p(\boldsymbol{\theta} \mid \mathcal{D})$.

As explained in Section 2.6.1, however, this theoretical possibility is not always useful in practice. As the size of $\mathcal{S}$ grows, computing the matrix exponential required for (4.1) becomes very computationally expensive — and this is a step that must be repeated for every sampled parameter value. An alternative to using (4.1) is to work directly with the CME. This involves constructing and numerically solving a set of ODEs whose number is equal to the size of the state-space. This method is complicated by the rapid combinatorial growth of the number of states in pMJPs, which leads to the need for optimisations such as in [6]; furthermore, the ODE solution can be numerically challenging due to stiffness. More importantly, these approaches fall short when $\mathcal{S}$ is infinite, in which case the generator cannot even be directly computed.

4.2 Inference for infinite state-spaces with the Finite State Projection

A simple approach to dealing with infinite state-spaces is to impose a limit on the count of each species, thereby creating a bounded state-space. This can be done, for instance, by using some heuristic about counts that the process is not likely to exceed, or by removing states that are not deemed likely enough, as in [6]. Such methods, however, introduce inaccuracies which are often hard to quantify. The Finite State Projection (FSP) [94] presents an alternative, systematic method of bounding the state-space in a way that maintains a desirable accuracy.

The FSP is a method for approximating the transient probability p_t of a MJP at a given time t and starting from a known initial distribution $\mathbf{p}_0$. A sketch of the method is shown in Algorithm 3. It works by constructing a truncation $\hat{S}$ of the state-space, so that the probability of escaping $\hat{S}$ during $[0,t]$ is less than a specified error threshold ε; the method computes an approximate probability distribution $\hat{p}_t$ by working on this reduced space. It can be applied in order to achieve performance benefits from reducing large state-spaces, or to make infinite state-spaces tractable by constructing a finite approximation to them. The resulting approximate distribution $\hat{p}_t$ can be used in the computations of the likelihood in (4.2) as a proxy for the true solution of the CME. Perhaps surprisingly, this approach does not seem to have been explored in a Bayesian setting, although there has been some work on using it for maximum likelihood parameter estimation [68].

The main issue is the construction of the space $\hat{S}$. The process is generally iterative, starting from a small set of states and adding more until the error condition is satisfied. However, this process can result in adding states that do not contribute much to the accuracy, but only grow the state-space. Checking whether the error condition is met after every expansion requires solving the CME repeatedly. Additionally, some implementations ([95]) suggest running the SSA to "seed" the initial set of states. All these issues compound the complexity of the method. A particularly troublesome point is that the space $\hat{S}$ and thus the solution $\hat{p}_t$ are obtained by analysing the process for specific values of its parameters. As different parameter values vary the dynamics, it cannot be assumed that the error guarantee reached will hold for the whole parameter space. It therefore becomes necessary to repeat the whole procedure whenever different parameters are considered, i.e. in every step of the M-H sampler.

4.3 Random truncations of the state-space

In a sense, the FSP tries to create an optimal (for the given error treshold) truncation of the state-space, but does not always manage to yield a minimal space. We now present an alternative approach which overcomes the drawbacks of the FSP, in particular creating unnecessarily large state-spaces. This is achieved by truncating the state-space *randomly*, while maintaining desirable statistical properties. We begin by describing the underlying method, long present in the field of computational physics [102] but recently introduced in the statistics and machine learning communities [80].

Algorithm 3 Finite State Projection (as in [94])

Input: Initial distribution $\mathbf{p}_0$, final time t, error threshold ε

Output: A truncated state-space $\hat{S}$ and approximate solution of the CME $\hat{\mathbf{p}}_t$

1: Initialise $\hat{S}$
2: Compute generator $\mathbf{A}$ for $\hat{S}$
3: Calculate $\hat{\mathbf{p}}_t = \mathbf{p}_0 e^{\mathbf{A}t}$, an approximation to the true solution of the CME
4: Compute error $= 1 - ||\hat{\mathbf{p}}_t||$, i.e. the probability of leaving $\hat{S}$ before t
5: **while** error $> \varepsilon$ **do**
6: Expand $\hat{S}$
7: Expand $\mathbf{A}$ by adding rows and columns corresponding to the new states
8: Compute $\hat{\mathbf{p}}_t$ and error for the new $\hat{S}$
9: **end while**
10: **return** $\hat{S}$, $\hat{\mathbf{p}}_t$

4.3.1 The Russian Roulette method

Assume we wish to estimate an infinite sum $a = \sum_{N=0}^{\infty} a_N$, where each term a_N is computable. One way of approximating the sum is to pick a single term a_k, where k is chosen from any discrete distribution with mass $p_0, p_1, \ldots$. We can immediately see that $\hat{a} = \frac{a_k}{p_k}$ has expectation $E[\hat{a}] = \sum_{N=0}^{\infty} \frac{a_N}{p_N} p_N = a$ and is therefore an unbiased estimator of the infinite sum. An issue with this approach is that, depending on the choice of distribution p_i, the variance of $\hat{a}$ might be very large, even infinite.

A reduced variance estimator can be obtained by approximating a with a partial sum up to order N, weighted appropriately. The number of terms is chosen randomly — at every term j, a random choice is made: there is a probability $1 - q_j$ of stopping the sum, otherwise we continue to form iteratively the partial sum $\hat{a} = \sum_{N=0}^{j} \frac{a_N}{p_N}$, where $p_N = \prod_{j=1}^{N-1} q_j$ (see Algorithm 4). The chance of definitely stopping this process at each iteration lends the scheme its *"Russian Roulette"* name. It can be shown (see, for example, [80]) that $E[\hat{a}] = a$, therefore the method yields an unbiased estimator of the full sum. Its variance can be shown to be finite under certain conditions; we return to this issue later in this section.

4.3.2 Truncating the likelihood

In order to use this method, we must express the likelihood as an infinite sum. The choice of what variable to sum over (in other words, marginalise out) is not clear.

Algorithm 4 Russian Roulette

Input: Terms a_n, stopping probabilities q_n

Output: An unbiased estimate $\hat{a}$ of $\sum_{n=0}^{\infty} a_n$

$\hat{a} \leftarrow 0$
$p \leftarrow 1$
$i \leftarrow 0$
stop $\leftarrow$ false
repeat
    $\hat{a} \leftarrow \hat{a} + \frac{a_i}{p}$
    $p \leftarrow p \cdot q_i$
    $i \leftarrow i + 1$
    with probability q_i, stop $\leftarrow$ true
until stop
return $\hat{a}$

Previous work ([17]) attempted to marginalise out the number of reactions that have occurred within a given time. However, this led to a method with complex calculations and very slow performance. Our approach, instead, is to marginalise out the maximum state reached in that time.

For simplicity, we start by considering a one-dimensional process with known initial state s_0 and a single observation point (t, s_t), measured noiselessly. The likelihood in this case can be written as:

$$p(s_t \mid s_0, \boldsymbol{\theta}) = \sum_{N=0}^{\infty} p_t(s_t, \max(s_{0:t} - s^*) = N \mid s_0, \boldsymbol{\theta}) \tag{4.3}$$

where $s_{0:t}$ represents the possible states visited during $[0,t]$, and $s^* = \max(s_0, s_t)$. Essentially, we are decomposing the space of process trajectories into a nested sum over subspaces of trajectories which differ by at most N from the observations. Note that the constraint on the maximum of $(s_{0:t})$ does not simply define a state-space, but constrains us to consider only those trajectories that actually achieve a "dispersal" of N. It is thus not exactly the solution of the CME. However, if we define

$$f_t^{(N)}(s', s) = p_t(s', \max(s_{0:t} - s^*) \leq N \mid s, \boldsymbol{\theta})$$

then each term in (4.3) can be decomposed as:

$$p_t^{(N)}(s_t, s_0) \equiv p_t(s_t, \max(s_{0:t} - s^*) = N \mid s_0, \boldsymbol{\theta}) = f_t^{(N)}(s_t, s_0) - f_t^{(N-1)}(s_t, s_0) \tag{4.4}$$

Each of the two terms in (4.4) is a transient probability on a finite state-space, and can be computed by constructing the corresponding generator matrix and solving the CME. Note, also, that through the two equations above, the likelihood is now expressed as a telescopic sum. This means that, when computing term $p^{(N)}$ in (4.3), the second sub-term of (4.4) has already been calculated for $p^{(N-1)}$. It is worth pointing out that computing $f_t^{(N)}(s_t, s_0)$ from $f_t^{(N-1)}(s_t, s_0)$ is not trivial and requires solving the CME again, even if the corresponding expansion of the state-space is small.

This is easily generalised to multi-dimensional processes by simply considering each dimension separately. For initial state $\mathbf{s_0} = (s_0^1, \ldots, s_0^M)$ and observation $\mathbf{s_t} = (s_t^1, \ldots, s_t^M)$, we have $\mathbf{s}^* = \big(\max(s_0^1, s_t^1), \ldots, \max(s_0^M, s_t^M)\big)$. The constraints we impose on $\mathbf{s}_{0:t}$ are then pointwise extended to each dimension, i.e. for $f_t^{(N)}$ we require $s^i - \max(s_0^i, s_t^i) \leq N$ for each $i = 1, \ldots, M$ during $[0,t]$, and for $p_t^{(N)}$ we require that $s^i - \max(s_0^i, s_t^i) = N$ for at least one i and some time point in $[0,t]$.

Example. Consider a two-dimensional process. Assume that we noiselessly observe its value at times 0 and t as $\mathbf{s}_0 = (1,1)$ and $\mathbf{s}_t = (0,2)$. From these two measurements, we form $s^* = (1,2)$ which will be the "base" of the state-spaces we construct, i.e. the state against which the dispersion is measured. We define the spaces $\mathcal{S}_0$ and $\mathcal{S}_1$ as shown in Figure 4.1. The terms $f_t^{(n)}(s, s_0)$ for $n = 0, 1$ correspond to the probability of being in state s at time t assuming the process has never left $\mathcal{S}_n$. The term

$$p_t^{(1)}(s_t, s_0) = f_t^{(1)}(s_t, s_0) - f_t^{(0)}(s_t, s_0)$$

is then the increase in the likelihood by considering the space $\mathcal{S}_1$ instead of just $\mathcal{S}_0$. Note that $f_t^{(n)}(s, s_0) = 0$ for $s \notin \mathcal{S}_n$, and consequently $p_t^{(n)}(s, s_0) = 0$.

It is also straightforward to incorporate noisy observations. If y is measured through a noise model $\pi(y \mid s)$, as above, then each summand in (4.3) is instead:

$$\begin{aligned} p_t^{(N)}(y, s_0) &= \sum_s \pi(y \mid s)\; p\left(s_t = s, \max\left(s_{0:t} - s^*\right) = N\right) \\ &= \sum_s \pi(y \mid s) \left(f^{(N)}(s, s_0) - f^{(N-1)}(s, s_0) \right) \end{aligned}$$

where now $s^* = \max(s_0, y)$. Although this is written as a sum over all possible states, the terms $f^{(N)}(s, s_0)$ and $f^{(N)}(s, s_0)$ will be zero for all s s.t. $s - s^* > N$. The computations can therefore proceed exactly as before.

Generalising to a whole time-series of observations $\mathcal{D} = \{(t_i, y_i)\}_{i=1}^N$, the Markovian and time-homogeneous nature of the process allows us to decompose the likelihood as:

$$p(\mathcal{D} \mid \boldsymbol{\theta}) = p_{\Delta t_1}(y_2 \mid y_1, \boldsymbol{\theta}) \cdots p_{\Delta t_{N-1}}(y_N \mid y_{N-1}, \boldsymbol{\theta})$$

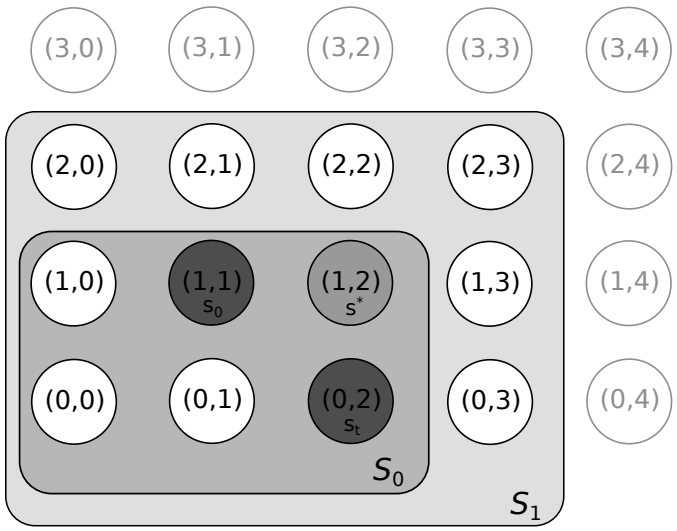


Figure 4.1: Example of truncated state-spaces, showing the observed states, the "base" state and the first two spaces $\mathcal{S}_0$ and $\mathcal{S}_1$.

where $\Delta t_i = t_{i+1} - t_i$. Note that, for each interval (t_i, t_{i+1}), the previous observation y_i induces an initial distribution over states by inverting the noise model $\pi(y_i \mid s)$[1]. An unbiased estimate $\hat{\mathcal{L}}_i$ for every factor can be computed as above. Since these are independent, $\hat{\mathcal{L}} = \prod_{i=1}^{N} \hat{\mathcal{L}}_i$ is an estimator for the whole likelihood and is still unbiased.

4.3.3 Metropolis-Hastings sampling with random truncations

We have described how to use the Russian Roulette procedure to obtain an unbiased estimate $\hat{L}(\mathcal{D} \mid \boldsymbol{\theta})$ of the likelihood. This can be used instead of the true likelihood in a pseudo-marginal M-H algorithm, in order to draw samples from the true posterior distribution over $\boldsymbol{\theta}$ (see Algorithm 5).

For the purposes of this chapter, we choose a q_n sequence such that the probability of accepting a term decreases geometrically; specifically, we use $q_n = aq_{n-1}$, with $q_0 = 1$ and $a = 0.95$. We show an empirical analysis of the variance in Section 4.5.1 that validates our choice of q_n and indicates that performance is robust with respect to the choice of the particular stopping distribution.

[1] We assume here that the inverted model has finite support, i.e. that the set $\{s \mid \pi(y_i \mid s) > 0\}$ of states from which an observation y_i is possible is finite.

Algorithm 5 Metropolis-Hastings sampler with random truncations

Input: Observations $\mathcal{D} = \{(t_i, y_i)\}_{i=1}^{N}$

Output: Samples $\{\boldsymbol{\theta}_i\}$ from the posterior distribution $p(\boldsymbol{\theta} \mid \mathcal{D})$

1: Sample parameters $\boldsymbol{\theta}_0$ from priors
2: Choose a truncation point randomly via the Russian Roulette procedure
3: Compute L, an estimate of likelihood as per Section 4.3.2.
4: $i \leftarrow 1$
5: **repeat**
6: Propose new parameters $\boldsymbol{\theta}^*$
7: Choose a truncation point randomly
8: Compute estimate of likelihood L^*
9: Compute acceptance probability $a = \min\left(\frac{L^*}{L}\frac{p(\boldsymbol{\theta}^*)}{p(\boldsymbol{\theta}_{i-1})}, 1\right)$
10: With probability a, set $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}^*$ and $L \leftarrow L^*$; else, $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_{i-1}$
11: $i \leftarrow i+1$
12: **until** convergence reached or enough samples taken

4.3.4 Variance of the estimator

An important issue with pseudo-marginal methods is that of the variance of the estimator used for the likelihood, as a high variance can have a severe impact on the performance of the sampler. In general, the estimators obtained via Russian Roulette can have infinite variance. However, we show conditions under which we can ensure that the variance is finite. Starting from the general result, we will use specific properties of the terms in (4.3) and interpret the conclusion in the MJP context.

The main result on the variance of the Russian Roulette-style estimator can be found in [80]:

Lemma 3. *Consider an estimator $\hat{a}$ of a sum $a = \sum_{N=0}^{\infty} a_N$, constructed as in 4.3.1. If*

$$\sum_{N=0}^{\infty} \frac{|a_N|}{p_N} \sup_{j \geq N} \left| \sum_{l=N}^{j} a_l \right| < \infty$$

then the variance of $\hat{a}$ is finite.

We first observe that in our case the summands a_N are probabilities and therefore non-negative, so $|a_N| = a_N$. This also lets us simplify the expression $\sup_{j \geq N} \left| \sum_{l=N}^{j} a_l \right| = \sup_{j \geq N} \sum_{l=N}^{j} a_l$: for a given N and different values of the upper limit j, we obtain the infinite sequence

$$\{a_N, a_N + a_{N+1}, a_N + a_{N+1} + a_{N+2}, \cdots\}$$

Since $a_i \geq 0$, we can see that this sequence is non-decreasing. Therefore,

$$\sup_{j \geq N} \left| \sum_{l=N}^{j} a_l \right| = \lim_{j \to \infty} \sum_{l=N}^{j} a_l = \sum_{l=N}^{\infty} a_l = a - \sum_{l=0}^{N-1} a_l \leq a$$

Since in this case a is a probability, we can conclude that

$$\sup_{j \geq N} \left| \sum_{l=N}^{j} a_l \right| \leq 1$$

Based on the above, the terms of the sum in Lemma 3 can be written as

$$v_N \equiv \frac{|a_N|}{p_N} \sup_{j \geq N} \left| \sum_{l=N}^{j} a_l \right| = \frac{a_N}{p_N} \left(a - \sum_{l=0}^{N-1} a_l \right)$$

Now consider the ratio between two consecutive terms:

$$\frac{v_{N+1}}{v_N} = \frac{a_{N+1}}{a_N} \frac{p_N}{p_{N+1}} \frac{a - \sum_{l=0}^{N} a_l}{a - \sum_{l=0}^{N-1} a_l}$$

where clearly

$$a - \sum_{l=0}^{N} a_l = a - \sum_{l=0}^{N-1} a_l - a_N \leq a - \sum_{l=0}^{N-1} a_l$$

Therefore

$$\frac{v_{N+1}}{v_N} \leq \frac{a_{N+1}}{a_N} \frac{p_N}{p_{N+1}}$$

We know that $\sum_{N=0}^{\infty} v_N$ will converge (and, from Lemma 3, the variance of the estimator will be finite) if $\frac{v_{N+1}}{v_N} < 1$; a sufficient condition for finite variance is therefore

$$\frac{a_{N+1}}{a_N} < \frac{p_{N+1}}{p_N}$$

This result suggests choosing the stopping distribution q_N so that the survival probability p_N decreases slower than a_N. In terms of MJPs, recall that a_N is the *additional* probability of reaching the target state if we expand the state-space. What is important, therefore, is (perhaps unsurprisingly) the asymptotic behaviour of the process as we consider increasingly larger state-spaces. To our knowledge, there have not been any results on this for either the general case or particular stochastic processes, and so we cannot offer advice on how to best choose the stopping distribution. However, the empirical results presented in this chapter show that a geometric distribution appears to work well and robustly for the examples considered.

4.4 Gibbs sampling

An alternative to the M-H sampler described so far has been explored by Rao and Teh in [109] and [110]. The main result of that work is a method for drawing sample paths from a MJP conditioned on observations $\mathcal{D}$. Furthermore, they show how this can be used as a step in a joint parameter-state Gibbs sampling algorithm. Their approach works with a formulation of the process in terms of exit rates and transition probabilities (cf. page 10).

Their idea is to place Gamma priors on each exit rate, and a Dirichlet prior on the vector of transition probabilities from each state. A simple analysis then shows that, conditioned on a full path of the process (S,T), the distribution of the exit rates is again Gamma, and that of the jump probabilities is Dirichlet, with easily computable hyperparameters. This is due to the exponentially-distributed waiting times. This result enables the construction of a Gibbs sampler with two steps: sampling a path (S,T) from $p(S,T \mid \boldsymbol{\theta}, \mathcal{D})$, and sampling parameters from $p(\boldsymbol{\theta} \mid S,T,\mathcal{D}) = p(\boldsymbol{\theta} \mid S,T)$. The last conditional independence relation simply reflects the fact that the full path contains more information than the observed points do.

While this idea is interesting, it has some drawbacks which limit its applicability. Firstly, considering each exit rate and jump probability vector separately quickly leads to a high number of dimensions. Additionally, it does not accurately reflect the truth in cases where the rates depend on shared parameters — as is, indeed, often the case with kinetic laws. Finally, the proposed method requires a finite state-space and is therefore inapplicable in open systems. This last point could be managed by imposing an upper limit, as described previously, but this can severely impact the performance of the algorithm, as we will show later.

We can keep the main idea of the approach and modify it to address the first two of these concerns. We will then show how it can be combined with a Roulette-style truncation strategy in order to manage infinite state-spaces.

4.4.1 Sampling a path

Before explaining our contributions, we summarise the algorithm proposed by Rao & Teh for sampling a path of a finite-state MJP from observations. This assumes that the process is fully specified, i.e. the rates of all its transitions are known.

The key idea is to impose a discretisation of time based on the principles behind uniformisation [62]. Essentially, any path (S,T) with transitions at times t can be

expanded to one that includes *virtual transitions* at times τ. On all virtual jump times between two real jump times t_i, t_{i+1}, the process loops back to state s_i. In [110], the authors make use of statistical properties of MJPs to show how the τ_i can be sampled so as to maintain the correct time-marginal distribution (essentially, that the number of virtual jumps is Poisson-distributed). Conditioned on the jump times (both true and virtual), the sequence of states can be viewed simply as a trace from a discrete-time process (one that allows self-loops). This discrete system can now be treated by standard methods: given a sequence of observations, a new sample of the process can be drawn through a forward filtering-backward sampling (FFBS) algorithm. This gives a new trace on the same times, which includes virtual jumps (but with the virtual jumps generally occurring at different times than in the original). Keeping only the true jumps results in new path from a continuous-time process. As shown in [110], this process produces paths drawn from the correct posterior process.

Algorithm 6 MJP Posterior Path Sampling (from [110])

Input: Observed data S at times T

Output: A path from the posterior process

1: Sample new times T' from a Poisson process
2: Augment times to $T^* = T \cup T'$
3: Run FFBS algorithm to get new state sequence V on T^*
4: Drop virtual jumps from V to obtain $\tilde{V}$ and corresponding times $\tilde{T}$
5: **return** $\tilde{T}, \tilde{V}$

4.4.2 Sampling the parameters

Instead of considering each transition independently from all others, we will instead assume that every transition is an instance of a *transition type*, reflecting the different kinds of interactions which can occur in the system. Each of the transition types has an associated rate law, which gives the rate of the transition as a function of the state (the correspondence with ProPPA reactions and kinetic laws is hopefully clear). These rate laws can depend on parameters. Each transition type also has an update vector $\mathbf{u}$, which determines the next state of the process: if occuring in state $\mathbf{s}$, then the next state is $\mathbf{s}+\mathbf{u}$. Since the number of transition types will generally be low, this significantly reduces the dimensionality of the problem, from quadratic in the number of states (with the Rao-Teh approach) to linear in the number of reaction types. Additionally, as the

parameters are shared by all transitions of a certain type, this accurately captures the dependence between the various transitions.

In order to still be able to take advantage of conjugacy, we impose the following three requirements on the transition types and their laws:

i. Each rate law has the form $f_i(s) = \theta_i \rho_i(s)$, where the $\rho_i(s)$ can be any function of the state.

ii. Each parameter θ_i has a Gamma prior: $\theta_i \sim \text{Gamma}(a_i, b_i)$.

iii. Each transition type has a distinct update vector.

Let (S,T) be a full trajectory of a process with R transition types, where $S = (s_0, s_1, \ldots, s_K)$ is the sequence of states at times $T = (t_0, t_1, \ldots, t_K)$. From assumption (iii) above, inspection of S is enough to determine what kind of transition occurred at each step; let u_k denote the transition type at time t_{k+1}, i.e. the one whose update vector is $s_{k+1} - s_k$. The total rate of exiting state s_k is $r_k = \sum_{i=1}^{R} \theta_i \rho_i(s_k)$. Since the waiting time between jumps is exponentially distributed in a MJP, this gives

$$p(t_{k+1} \mid t_k, s_k) = r_k e^{-\Delta t_k r_k} \text{ , where } \Delta t_k = t_{k+1} - t_k$$

The probability of the next state being s_{k+1} is $\frac{\theta_{u_k} \rho_{u_k}(s_k)}{r_k}$. The total probability of the path is then:

$$\begin{aligned} p(S,T \mid \boldsymbol{\theta}) &= p(S \mid \boldsymbol{\theta}) p(T \mid S, \boldsymbol{\theta}) \\ &= \prod_{k=0}^{K-1} \frac{\theta_{u_k} \rho_{u_k}(s_k)}{r_k} r_k e^{-\Delta t_k r_k} \\ &= \prod_{k=0}^{K-1} \theta_{u_k} \rho_{u_k}(s_k) e^{-\Delta t_k r_k} \end{aligned}$$

Since every parameter is Gamma-distributed *a priori*,

$$p(\theta_i) = \frac{b_i^{a_i}}{\Gamma(a_i)} \theta_i^{a_i - 1} e^{-b_i \theta_i}$$

we have:

$$\begin{aligned} p(\theta_i \mid S,T) &\propto p(\theta_i) p(S,T \mid \boldsymbol{\theta}) \\ &\propto \theta_i^{a_i + N_i - 1} e^{-b_i - \sum_{k=0}^{K-1} \Delta t_k \rho_i(s_k) \theta_i} \end{aligned} \tag{4.5}$$

Therefore, conditioned on the full trace, the parameters are again Gamma-distributed with shape $a_i + N_i$ and rate $b_i + \sum_{k=0}^{K-1} \Delta t_k \rho_i(s_k)$, where N_i is the number of times the ith

transition type is observed in the trace (a similar derivation can be found in Chapter 10 of [130]). Hence, we have *exact* Gibbs updates for the kinetic parameters; notice that, since we have a single parameter for each reaction, the number of parameters to be sampled is often orders of magnitude lower than the number of parameters sampled in [110] (one per possible state transition), yielding computational and storage savings.

4.4.3 Modified Gibbs with random truncation

The algorithm described above can be modified so as to allow for infinite state-spaces. The difficulty is that there is no direct way to sample trajectories without a bound on the state-space, as the uniformisation sampler requires a finite number of states. To work around this limitation, we propose to sample a truncation point via the Russian Roulette procedure. This defines a finite state-space, on which we can then draw a trajectory and parameters as above. Essentially, we are introducing an auxiliary variable in the sampler, the truncation point m^*. The benefit is that conditioning on m^* lets us sample paths and parameters as before. The drawback, however, is that we are no longer considering the correct conditional distributions required for Gibbs sampling, i.e. we sample from $p(S,T \mid \mathcal{D},\theta,m^*)$ instead of $p(S,T \mid \mathcal{D},\theta)$. We are therefore no longer able to accept every trajectory and parameter sample drawn, and must instead introduce an acceptance ratio. We thus refer to this as a modified Gibbs or Gibbs-like algorithm. A sketch of the main part is given in Algorithm 7.

Algorithm 7 Modified Gibbs sampler with random truncation (sampling step)

Input: Observations $\mathcal{D}$, current parameters $\boldsymbol{\theta}_i$ and path (S_i,T_i)

Output: Next sample of parameters $\boldsymbol{\theta}_{i+1}$ and path (S_{i+1},T_{i+1})

1: Propose $\boldsymbol{\theta}^*$ from $p(\boldsymbol{\theta} \mid S_i,T_i)$ as per Equation (4.5)
2: Choose a truncation point m^* via Russian Roulette
3: Propose a path (S^*,T^*) from $p(S,T \mid \mathcal{D},\boldsymbol{\theta}^*,m^*)$ as per Section 4.4.1
4: Compute the acceptance ratio a from Equation (4.6)
5: With probability $\min(a,1)$, set $\boldsymbol{\theta}_{i+1} \leftarrow \boldsymbol{\theta}^*$ and $(S_{i+1},T_{i+1}) \leftarrow (S^*,T^*)$; else set $\boldsymbol{\theta}_{i+1} \leftarrow \boldsymbol{\theta}_i$ and $(S_{i+1},T_{i+1}) \leftarrow (S_i,T_i)$

The acceptance ratio at every step is

$$a = \frac{p^{(i+1)}(S^* \mid \boldsymbol{\theta}^*,\mathcal{D})p^{(i+1)}(S_i \mid \boldsymbol{\theta}^*,\mathcal{D})}{p^{(i)}(S_i \mid \boldsymbol{\theta}^*,\mathcal{D})p^{(i)}(S^* \mid \boldsymbol{\theta}^*,\mathcal{D})} \tag{4.6}$$

where $p^{(n)}(S \mid \boldsymbol{\theta}^*,\mathcal{D})$ is the posterior probability of a trajectory S under the truncation

point in iteration n. We must therefore compute the probabilities of the "old" and proposed paths under both the previous and the currently proposed truncation.

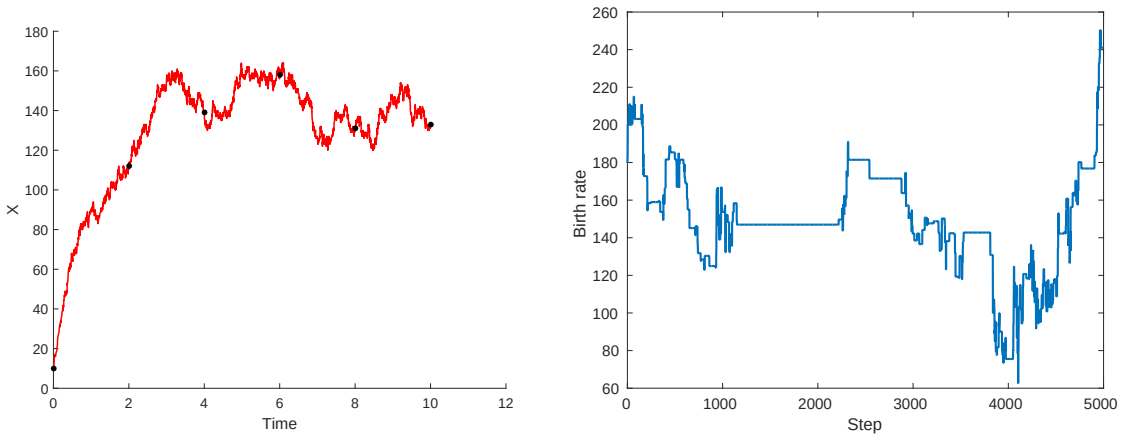
These quantities can be computed via the forward-backward algorithm on the corresponding state-spaces. Note that, if we could draw trajectories from the whole state-space without truncating it, the terms in a would cancel out, giving standard Gibbs sampling with acceptance rate of 1.

It is important to observe that this auxiliary variable Gibbs sampler actually targets the joint posterior distribution of parameters and trajectories. As such, it provides richer information than the M-H sampler (which directly targets the parameter posterior), but may be less effective if one is solely interested in parameter inference. The performance can also be affected by computational factors, particularly the costs of drawing sample trajectories (which is not needed if we compute the likelihood by matrix exponentiation). In general, such costs will be model- and data-dependent, so that some domain knowledge or initial exploration may be advisable before deciding which of the two proposed algorithms to use.

4.5 Example results

4.5.1 Empirical variance

Before showing how our algorithms perform against the state of the art, we present empirical evidence that our Russian Roulette-style truncation approach produces estimators with low variance, an issue that has recently received attention in pseudo-marginal methods [28, 119]. Additionally, we show that the estimator is robust to the choice of the particular stopping distribution q_n used in the truncation scheme. To verify this, we considered three different q_n sequences, applied to the predator-prey model described in Section 4.5.2. All schemes were of the form $q_n = aq_{n-1}$ with $q_0 = 1$ and $a \in \{0.95, 0.75, 0.2\}$, respectively yielding 5.6, 2.4 and 1.2 terms on average. For each scheme, we calculated 1000 estimates of the transition probabilities between observations, obtaining estimates of the log-likelihood and computing its mean and variance. This was repeated for 10 different parameterizations of the model. It can be seen (Table 4.1) that the variance of the estimator (measured as the coefficient of variation of the log-likelihood) is consistently low. This validates our approach and indicates that the stopping distribution does not critically affect performance and therefore does not require fine-tuning. An intuitive explanation for this robustness is that the state-spaces



(a) (b)

Figure 4.2: (a) Full trace (continuous line) and observations (dots) used in the birth-death process example; (b) Parameter samples for the birth rate using Algorithm 1, illustrating undesirable "sticking" behaviour when taking 5.6 terms on average

considered must include the observations by construction, and therefore the base space already contains significant probability mass. Taking even a single term thus gives a good approximation of the likelihood, unless the observations are too sparse or the process is very "volatile" and therefore very likely to exceed the observed values.

As an example of the limitations of our approach, we considered the example of a birth-death process involving a single species, X, with a constant birth rate of 150 and a death rate of X. From an initial value of $X = 10$, we simulated the system and used the values at 5 time points (Figure 4.2a). The three truncation schemes described above did not yield accurate estimates, even when taking 5 terms on average. With a stopping scheme $q_n = 0.99q_{n-1}$ (corresponding to 12.5 terms on average), we were able to get good estimates of the true probabilities. The more aggressive truncation schemes display higher variance and could cause problems when their estimates are used in Algorithm 1: when taking 5.6 terms on average, the variance causes the sampling chain to "stick", as seen in Figure 4.2b. We will return to this issue at the end of this chapter.

4.5.2 Benchmark data sets

We now assess the performance of the two algorithms described in the previous section as well as the simple Gibbs sampler based on uniformisation. We could not run the original Gibbs sampler of [110] as the high number of parameters (one per state) swiftly led to storage problems; instead, we used our variant of it for finite systems, as described

Configuration	$a = 0.95$ (5.6 terms)	$a = 0.75$ (2.4 terms)	$a = 0.2$ (1.2 terms)
1	0.0002	0.0008	0.0223
2	0.0051	0.0151	0.0344
3	0.0003	0.0013	0.0245
4	$< 10^{-4}$	$< 10^{-4}$	0.0016
5	$< 10^{-4}$	$< 10^{-4}$	0.0008
6	0.0005	0.0021	0.0109
7	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$
8	0.0003	0.0014	0.0223
9	0.0002	0.0011	0.0073
10	0.0002	0.0009	0.0077

Table 4.1: Coefficient of variation for the log-likelihood, estimated from 1000 samples for the LV model, under three truncation schemes (varying α) and ten parameter configurations (Section 4.5.1)

in Sections 4.4.1 and 4.4.2. We first compared the performance of the three methods on two widely used pMJP models:

Lotka-Volterra (LV) model This predator-prey system involves four types of reactions, representing the birth and death of each species, and is a classic model in ecology and biochemistry. Truncated LV processes have been studied in previous work ([100],[17]), making it an attractive candidate for evaluating our approach.

$$\begin{aligned} X + Y &\to 2X + Y \quad \text{at rate } \theta_1 XY \\ X &\to \varnothing \quad \text{at rate } \theta_2 X \\ Y &\to 2Y \quad \text{at rate } \theta_3 Y \\ X + Y &\to X \quad \text{at rate } \theta_4 XY \end{aligned}$$

We start from an initial state of 7 predators and 20 prey. When a finite state-space is required, we impose a maximum count of 100 for each species, as in previous work. For completeness, a ProPPA model of this system is shown in Figure 4.3.

SIR epidemic model A commonly-used model of disease spreading (see e.g. [5]), where the state comprises three kinds of individuals: S(usceptible), I(nfected) and

```
a = Gamma(4,10000);
b = Gamma(4,10000);
c = Gamma(4,10000);
d = Gamma(4,10000);

kineticLawOf birthPred: theta1 * X * Y;
kineticLawOf deathPred: theta2 * X;
kineticLawOf birthPrey: theta3 * Y;
kineticLawOf deathPrey: theta4 * X * Y;

X = birthPred ↑ + deathPred ↓ + deathPrey ⊕;
Y = birthPrey ↑ + deathPrey ↓ + birthPred ⊕;

X[7]<*>Y[20]

observe(obsPredPrey);
infer(rouletteMH);
```

Figure 4.3: ProPPA description of the LV model.

R(ecovered). We examine two variants of the model, a finite version where the total population is constant:

$$\begin{aligned} S+I &\rightarrow 2I \quad \text{at rate } \theta_1 SI \\ I &\rightarrow R \quad \text{at rate } \theta_2 I \end{aligned}$$

and an infinite state variant where new individuals can join the S population with unknown arrival rate:

$$\varnothing \rightarrow S \quad \text{at rate } \theta_3$$

The initial state in both cases is $(S,I,R) = (10,5,0)$. For the finite-state version, this gives a state-space of 121 states. For the infinite case, we chose a truncation with upper limit $(28,33,33)$, corresponding to 18 new arrivals in the system. To see this, note that the number of arrivals in a time interval of duration T is Poisson-distributed, with mean $\theta_3 T$. We used the final observation time and the prior mean of θ_3, and chose the 95-percentile of the distribution governing the new arrivals. In broad terms, this means our truncation will accommodate new arrivals with 95% probability.

Table 4.2 summarises our evaluation results across the models considered; the metrics we use are total computational time for 5000 samples, mean relative error in parameter estimates (using the posterior mean as a point estimate), Effective Sample Size (ESS) per minute of computation, and number of iterations to convergence, defined as Potential Scale Reduction Factor (PSRF) < 1.1 [38].

Results on the LV model show that methods based on random truncations achieve very considerable improvements in performance compared to the simple Gibbs sampler (where the state space was truncated at a maximum number of 100 individuals per species). In particular, the Gibbs-like Roulette algorithm shows excellent behaviour in most aspects, with a high ESS suggesting it is a more efficient sampler. Its running time is comparable to that reported for a variational mean field approximation in [100], and its rapid convergence time suggests that this is a very competitive algorithm in practice. Sample results from that algorithm are presented in Figure 4.4 for the reaction parameters, and in Figure 4.5 for the state of the process itself. The M-H Roulette algorithm, while still computationally feasible, requires a long time to converge, reflecting potential difficulties in choosing effective proposal distributions (a problem naturally bypassed by the Gibbs-like approach). The simple Gibbs algorithm is much slower than the other two, undoubtedly owing to its large state-space of 10000 states and very high memory requirements during the FFBS algorithm (notice that the algorithm had to be run on a more powerful machine).

		Gibbs	Roulette M-H	Roulette Gibbs
LV	Time	1011min*	72min	**30min**
	Error	14%	11%	**10%**
	ESS/min	0.63*	0.5	**5.5**
	Steps to conv.	**24**	1314	180
SIR finite	Time	**2min**	11min	6min
	Error	3.66%	11.72%	**2.13%**
	ESS/min	**2150.34**	54.98	264.36
	Steps to conv.	**13**	33	27
SIR infinite	Time	1585min*	**291min**	666min*
	Error	31.6%	25%	**24.3%**
	ESS/min	0.45*	**2.6**	0.23*
	Steps to conv.	**5**	65	136

Table 4.2: Performance of the various algorithms tested. Metrics are averaged over all parameters. Where possible, experiments were performed on an Intel i3-2100 3.10GHz processor. Cases marked with * were tested on a 24-core Xeon E5-2680 2.5GHz due to increased memory requirements. Best results for each experiment and metric are shown in bold.

Note that the impact of the (necessarily) large truncation is twofold. Firstly, the large state-space directly affects the running time of the FFBS algorithm, whose complexity is quadratic in the number of states. Secondly, since the rates in this model are increasing functions, having states with high counts means the generator matrix has high diagonal entries (exit rates). This, in turn, requires choosing a high exit rate for uniformisation, leading to long paths with many self-jumps, and ultimately further slowing down the FFBS step. The results for this model clearly show the usefulness of the random truncation approach compared to using a static, conservative truncation.

Results on the SIR model show that, in the finite state-space case, the original Gibbs sampler is highly efficient and by some way the best algorithm. This is unsurprising, as truncations incur additional computational overheads which are not needed for such a small state space. The picture is completely different for the infinite SIR model. In this case, the M-H sampler clearly seems to be the best algorithm, achieving very fast convergence and outperforming the other two. For parameter values within the prior range, the infinite SIR model exhibits fast dynamics which lead to very long uniformised trajectories, considerably increasing the computational costs of sampling trajectories via the FFBS algorithm. The problem is further compounded for the simple Gibbs sampler. Even with the truncation described above, there are 32594 states, resulting in very severe computational and storage costs.

4.5.3 Genetic toggle switch

As a more realistic application of our approach, we consider a model of a synthetic biological circuit describing the interaction between two genes (G1 and G2) and the proteins they encode (P1 and P2). Each protein acts as a repressor for the other gene, inhibiting its expression. This leads to a bistable behaviour, switching between a state with high P1 and low P2, and one with low P1 and high P2 (hence the name *toggle-switch*). The interactions are encoded as eight chemical reactions (see also Figure 4.6):

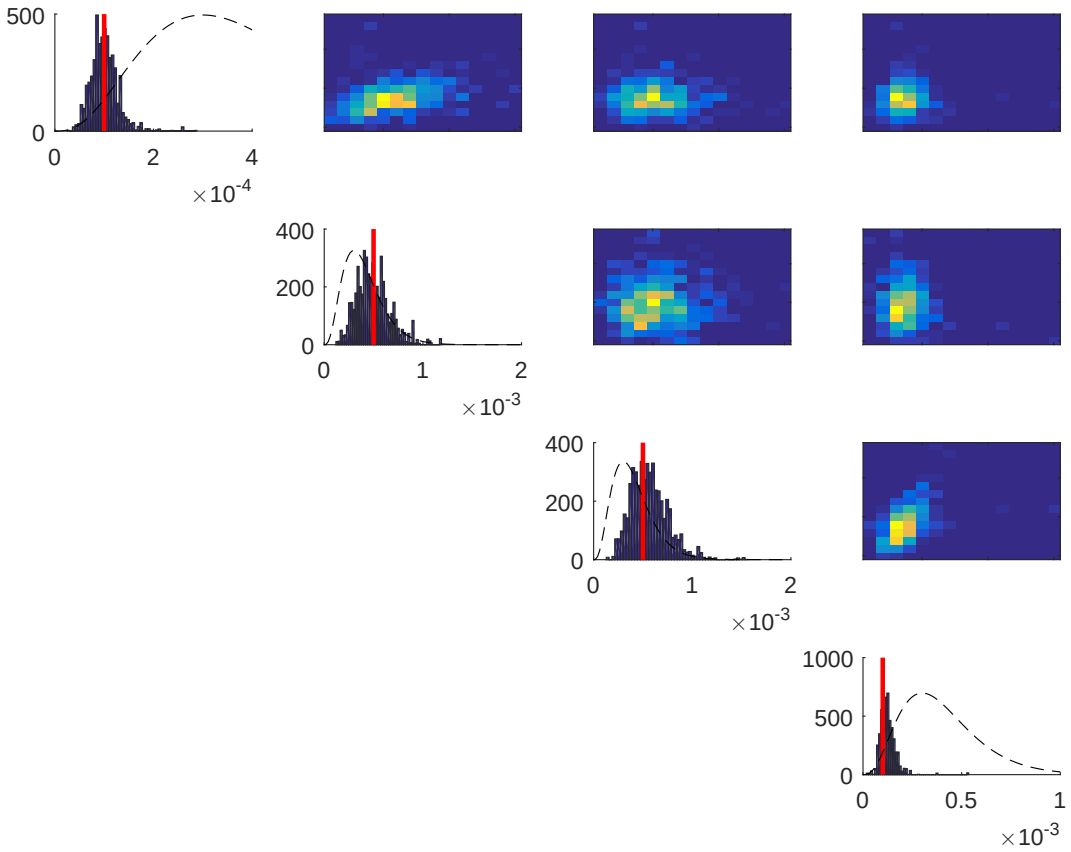


Figure 4.4: Posterior marginals and pairwise correlations for the parameters of the LV model, from 5000 samples using the Gibbs-like algorithm with random truncations (true values marked by red line, prior shown in dashed line).

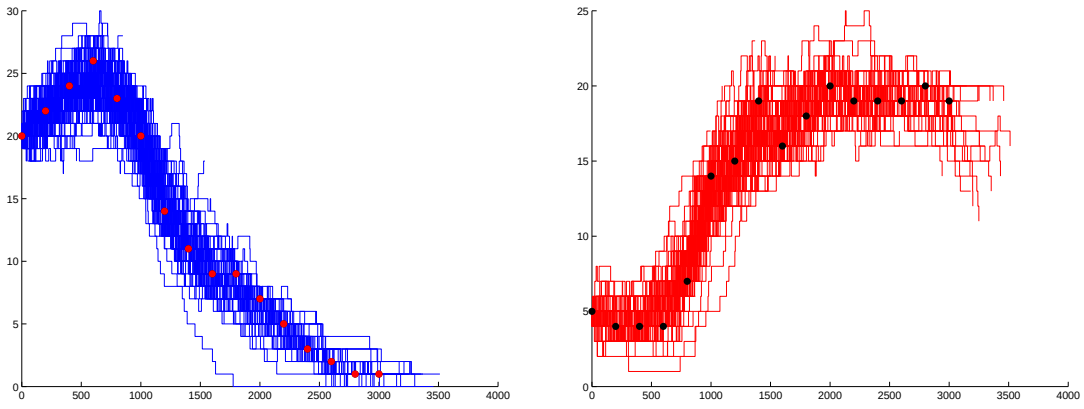


Figure 4.5: Samples of the posterior process for the predator-prey model, drawn using the Gibbs-like algorithm with random truncations: prey (left), predators (right). The dots indicate the observations used.

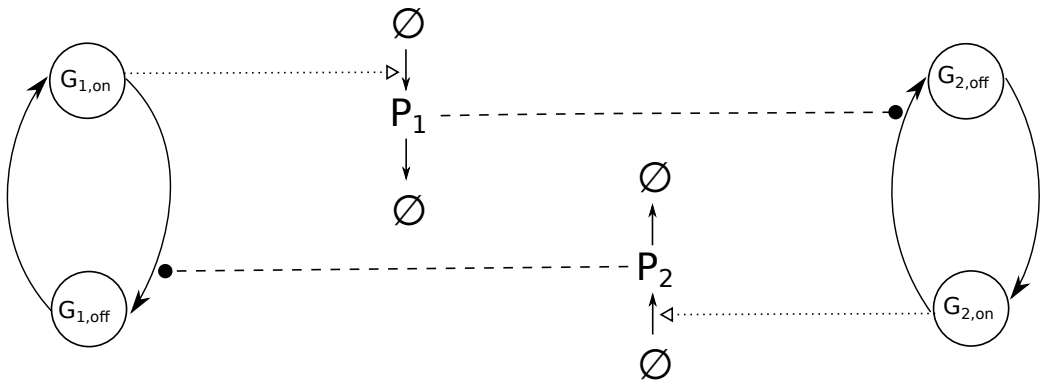


Figure 4.6: Depiction of the reactions and species involved in the genetic toggle switch example. Dashed lines indicate that a species accelerates a reaction. Dotted lines indicate that a species is required for a reaction.

$$
\begin{array}{lcll}
G_{1,on} & \rightarrow & G_{1,on} + P_1 & \text{at rate } \theta_1 \\
G_{2,on} & \rightarrow & G_{2,on} + P_2 & \text{at rate } \theta_2 \\
P_1 & \rightarrow & \varnothing & \text{at rate } \theta_3 P_1 \\
P_2 & \rightarrow & \varnothing & \text{at rate } \theta_4 P_1 \\
G_{1,off} & \rightarrow & G_{1,on} & \text{at rate } \theta_5 \\
G_{2,off} & \rightarrow & G_{2,on} & \text{at rate } \theta_6 \\
G_{1,on} & \rightarrow & G_{1,off} & \text{at rate } \theta_7 e^{rP_2} \\
G_{2,on} & \rightarrow & G_{2,off} & \text{at rate } \theta_8 e^{rP_1}
\end{array}
$$

where r is a constant assumed known.

This system was engineered *in vivo* in one of the pioneering studies in synthetic biology [37] and has been further studied in [123]. Statistical inference is increasingly being recognised as a crucial bottleneck in synthetic biology: while genome engineering technologies enable researchers to reliably synthesise circuits with a desired structure, predicting the dynamic behaviour of a circuit requires knowledge of the kinetic parameters of the system once it is implanted *in the cell*, which cannot be directly measured. As synthetic biology is intrinsically at the single cell level, inference techniques for stochastic models have the potential to be of great assistance in the rational design of synthetic biology circuits.

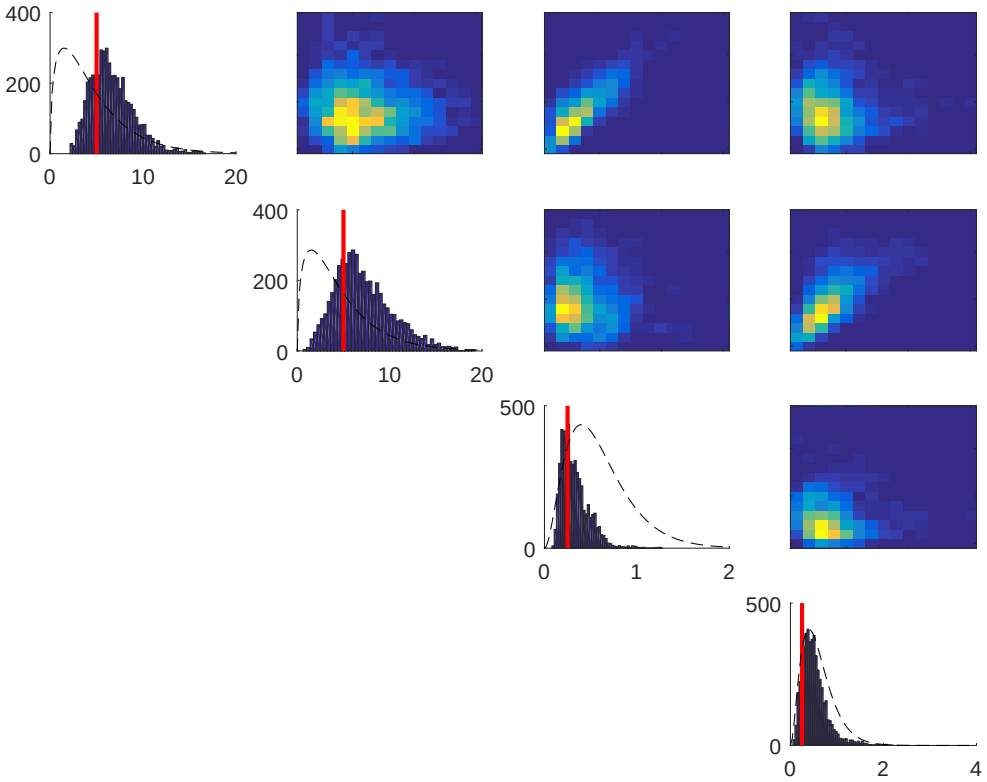
Following [123], we model the system using a binary state for each gene and discrete levels for the proteins. The genes can be active or inactive, with protein being produced only in the former case. Each gene can be modelled with a telegraph process (as in e.g. [114]): an inactive gene becomes active at a constant rate, and an active one

becomes inactive at a rate depending on the level of its repressor. When a gene is active, the level of its product follows a birth-death process; that is, proteins are produced at a constant rate and degrade at mass-action rates. We use a single production reaction for each protein to abstract various underlying mechanisms, including transcription and translation. The model comprises eight types of reaction; note that the requirements of our method on the form of the kinetic laws (Section 4.4.2) are flexible enough to accommodate the deactivation dynamics used here, even though they are not mass-action.

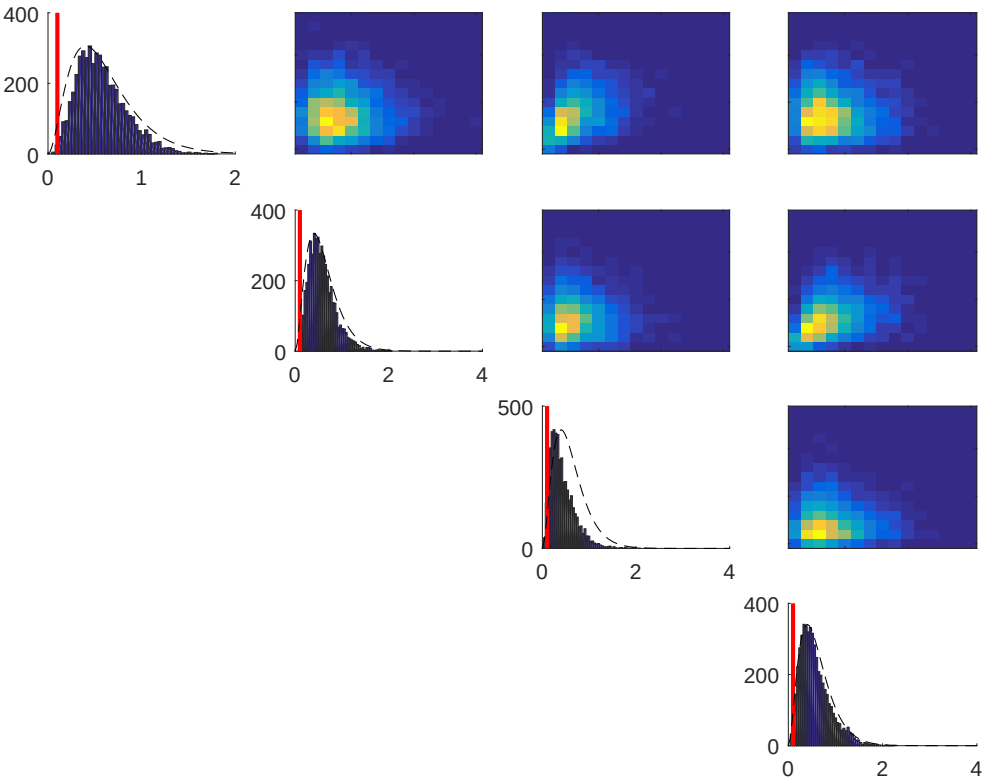
We used the Gibbs-like sampler to infer the joint posterior distribution of the eight parameters and state trajectories in this system. Our results indicate that the likelihood is relatively insensitive to the parameters governing the activation and deactivation of the two genes $(\theta_5$–$\theta_8)$. This is a reasonable result, since we do not observe the state of the genes but only the levels of the two protein products (although also note that the priors do not favour the true values of these parameters). Therefore, the effect of the switching parameters is seen only indirectly through the switching events, which are rare in the data. In contrast, the protein expression and degradation rates $(\theta_1$–$\theta_4)$ have sharp posteriors which capture interesting correlations between the parameters — for instance, we observe a strong correlation between the production and degradation rate of each protein, as perhaps expected given the similarity to a birth-death process. Figures 4.7 and 4.8 show parameter posteriors and convergence statistics for one such experiment, showcasing the good behaviour of the algorithm.

4.6 Discussion

This chapter has introduced two algorithms for inference in pMJPs that can handle systems with infinite state-spaces. From our experiments, it is not straightforward to say which of the two performs better. The second sampler has the potential to explore the parameter space more efficiently thanks to its Gibbs-like structure. Additionally, even though this was not used in this chapter, it produces sample paths from the posterior process, and can therefore directly be used for state inference. However, its inherent assumptions mean that it can only be applied to a restricted set of systems, and its performance can suffer from high memory consumption. On the other hand, the simpler pseudo-marginal M-H algorithm is more flexible, being generally applicable to any pMJP. Its longer running time in some of the experiments is due to the complexity of repeated matrix exponentiations, particularly when large truncations were involved.



(a)



(b)

Figure 4.7: Posterior marginals and pairwise correlations for the parameters of the toggle switch model, from 5000 samples using the Gibbs-like algorithm with random truncation (priors shown in dashed line): (a) θ_1, θ_2, θ_3 and θ_4; (b) θ_5, θ_6, θ_7 and θ_8.

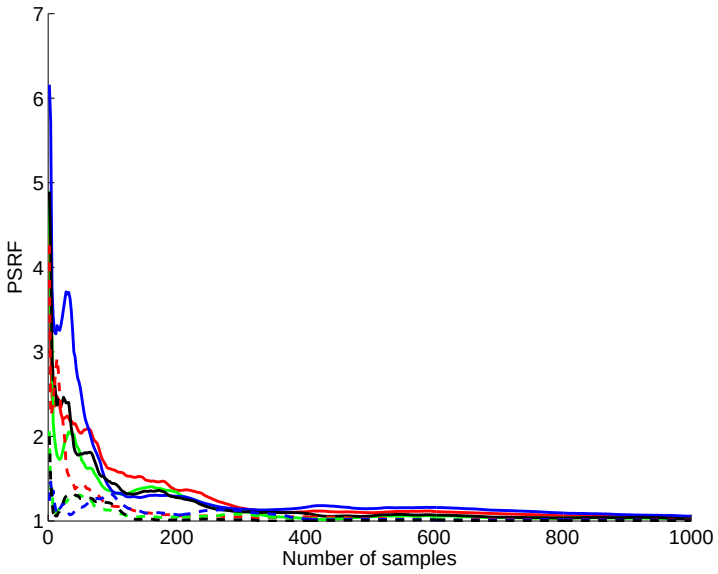


Figure 4.8: PSRF for all eight parameters of the toggle switch model, for the experiment of Figure 4.7.

As discussed previously, large state-spaces also naturally affect the performance of the Gibbs-like algorithm, especially when the rate functions lead to high discrepancy among the transition rates in the state-space. This is a consequence of the combined use of the FFBS algorithm and uniformisation. That said, the random truncation strategy employed in both algorithms offered increased performance compared to imposed truncations of the state-space.

The question of estimator variance is an important one in pseudo-marginal methods. We have presented empirical results for this issue rather than a full theoretical treatment, as the latter is not possible for the general case and would instead require analysing a particular system of interest. Our results, along with the intuition behind them, suggest that the truncation strategy is fairly robust to the stopping distribution used. However, the performance can degrade when the underlying process changes too much between observations, i.e. when observations are too sparse. This situation also arises when the populations involved are very high, in which case methods based on continuous approximations of the state (Section 2.6.4) will be more efficient. This means that our method is particularly suited to population sizes that are too large for direct likelihood computation (even with an imposed truncation) but not high enough to make continuous approximations applicable.

Chapter 5

Case study and implementation of the ProPPA tool

Chapter 3 introduced the ProPPA language but did not provide much information on how inference can be performed, while Chapter 4 presented novel algorithms for that purpose. The current chapter has two parts. The first goes into more detail about the software implementation of ProPPA, including the different algorithms that make up its core engine. The second part is a case study of real epidemiological data, which serves as an example to demonstrate the capabilities of the framework.

5.1 The ProPPA tool

The ProPPA tool was developed with two broad purposes in mind. First, to automate the inference process for user-specified models written in ProPPA. Second, to provide various utilities for analysing the behaviour of uncertain stochastic systems of the kind described in this thesis. It is therefore intended to be used both as a command-line tool and as a library of useful methods, but the presentation here focuses on the first goal.

Figure 5.1 summarises the various steps of processing a model.

5.1.1 Parsing

The first step is parsing the text file containing the ProPPA model. A model generally follows the syntax outlined in the language definition in Chapter 3. Although the semantics was presented there for any distribution and rate law, the implemented tool is restricted in what expressions it recognises. Specifically, it supports a finite set of

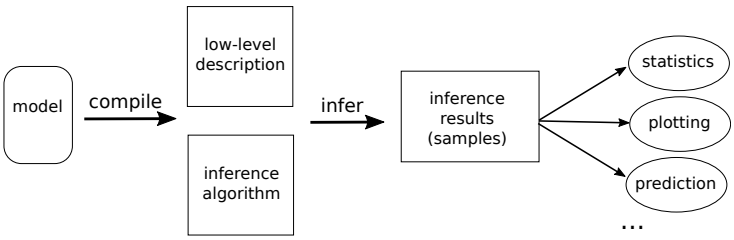


Figure 5.1: Diagram of the different steps for inferring the parameters of a ProPPA model: the model is compiled to extract a low-level representation of the system, the chosen inference algorithm is applied and returns a set of samples from the posterior parameter distribution.

families of distributions as priors for uncertain parameters, namely Uniform, Gaussian, Gamma and Exponential. The numerical expressions used to define rates can consist of any combination of basic operations, including exponentiation. Furthermore, additional functions are recognised that allow the tool to support more expressive models. These include the floor function and the Heaviside or step function

$$H(x) = \begin{cases} 0 \text{ if } x < 0 \\ 1 \text{ if } x > 0 \end{cases}$$

The textual representation of a model written for the tool uses slightly different notation than the formal syntax presented in Chapter 3. In particular, the $\uparrow$ and $\downarrow$ role indicators in species definitions are written as `>>` and `<<`, respectively. Similarly, $\oplus$, $\ominus$ and $\odot$ are written as `(+)`, `(-)` and `(.)`, and $\underset{*}{\bowtie}$ as `<*>`. Additionally, the sthoichiometry of a species can be omitted if it is 1, simplifying the description.

5.1.2 Compiling

Once parsed, the model is compiled into a lower-level representation. The main step is creating executable functions representing the rate law of each reaction. These functions accept a parameterisation and a state (vector of integers) and return a numerical rate. For example, a rate declaration of the form

$$\texttt{kineticLawOf}\ a : k * A$$

will be converted to a (Python) function `f`, such that `f(p)(s)` computes the expression kA, with the values for the parameter k and the count of component A retrieved from the lists `p` and `s`, respectively. To this end, the parsed expression is transformed back

into a textual representation that can be used as the body of a function declaration. A function is then declared with a dynamically generated body, and this is repeated for all reaction rate expressions. This process can be performed once and the resulting functions are available for any number of evaluations and any parameterisation. The efficiency of this approach is important, considering that the reaction rates must be repeatedly recalculated during the inference process.

Additionally, an update vector is extracted for each reaction by examining the definitions of each species. These are then combined into a stoichiometry matrix which shows how the quantity of each species is affected by each reaction. Essentially, the goal is to obtain a representation of the model as a pCTMC. This representation is what the inference algorithms act upon. The information extracted at this stage is also sufficient to construct the state-space of the process and its generator matrix (assuming it is finite); these may be required by some of the inference methods. This stage includes loading the observations, which are kept as a list of (time,state) pairs, generating the appropriate prior distribution for each uncertain parameter and other necessary book-keeping.

The main way through which the model itself communicates with the rest of the tool is the inference process, but other methods are also available for interaction. For instance, it can be made concrete by specifying values for some or all of the uncertain parameters. Figure 5.2 is a simplified diagram of the tool's architecture showing the different entities involved.

5.1.3 Samplers

The different algorithms are applied to the low-level representation obtained as described above. The appropriate tool, as specified by the `infer` statement of the model, is initialised using this representation and according to some default settings. If the user wishes, they can control the behaviour of the sampler by changing some of its parameters in a configuration file. This file is specified using an optional `configure` statement. The particular options available for tuning depend on the particular algorithm; generally, aspects that can be controlled include the variance of the proposal distributions for a parameter, observation noise, and number of samples to take.

All the methods included are sampling algorithms and (with the exception of the Gibbs ones) use Gaussian proposals. The different algorithms vary in their assumptions and applicability, as well as their inner working. We now briefly discuss the algorithms that form the core inference engine, along with their strengths and weaknesses; the decision of which to use depends on the comparative benefits of each method.

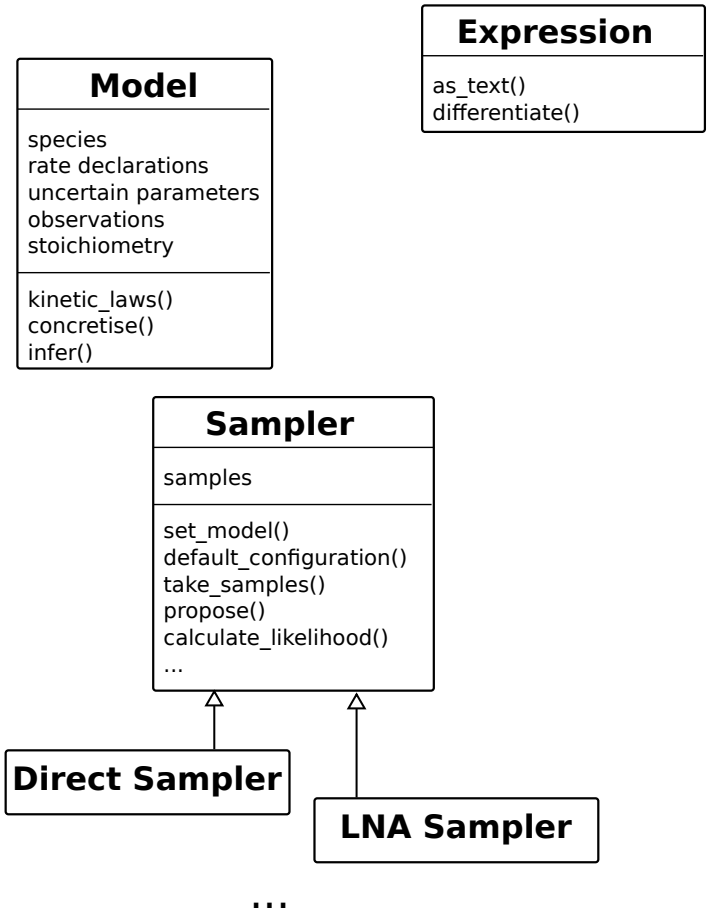


Figure 5.2: Basic architecture of the ProPPA implementation, showing the contents of the main classes and the interface they expose.

Algorithm	Exact	Infinite	Notes
`direct`	Yes	No	
`gibbs`	Yes	No	Requires Gamma priors and unique updates
`rouletteMH`	Yes	Yes	
`rouletteGibbs`	Yes	Yes	Requires Gamma priors and unique updates
`ABC`	No	Yes	Sensitive to chosen distance threshold
`fluid`	No	Yes	More accurate for systems with large counts
`LNA`	No	Yes	More accurate for systems with large counts, requires differentiable functions

Table 5.1: Summary of the available algorithms in the inference engine.

Direct solution

The simplest algorithm we include works by direct computation of the likelihood, which is feasible (at least theoretically) if the state-space of the system is finite. It is based on the analysis in Section 4.1, which involves explicitly computing the generator for each sampled set of parameters and exponentiating it. This is then used to compute the likelihood as part of a standard M-H sampler. Its cost scales badly with the number of states because of the matrix exponentiation, therefore this method is only advisable for small examples.

Gibbs sampling

Another method included in our framework uses Gibbs sampling, based on the method of Rao & Teh presented in Section 4.4. This means that it does not require matrix exponentiation, which makes it more suitable for models with larger spaces — it is still only applicable to finite systems, however. We have modified it slightly by adapting it to ProPPA-style descriptions of systems. As explained previously, the algorithm cannot be applied to all systems. Specifically, the kinetic law of each reaction must have the form $k_i f_i(s)$ where k_i is a parameter with a Gamma distribution and $f_i(s)$ is an arbitrary function of the state. Note that this includes, but is not limited to, mass-action laws. It is additionally required that each reaction have a distinct update vector.

Approximate Bayesian Computation

ProPPA can model systems of different size, including ones with infinite state-space. In the latter case, the two previous methods cannot be used; even for large finite spaces, their performance can be unsatisfactory. One of the methods we include works around this limitation by following the ABC approach [125], the use of which was illustrated in Section 3.5.3. ABC requires only simulating the system, which makes it a good fit for CTMCs. For every parameter sampled, instead of computing the likelihood, the system is simulated via Gillespie's algorithm (which is straightforward to do from the extracted representation). The trace obtained is then compared to the observations using a distance metric. If this distance is greater than a threshold ε (specified by the user in the configuration file), the sample is rejected; otherwise, it is accepted with a probability similar to the one in the usual M-H acceptance ratio. The parameter samples returned are drawn from an approximation to the posterior, one that converges to the true distribution as $\varepsilon \to 0$. The accuracy of ABC depends on the value chosen for ε, but

its simplicity means that it does not impose any constraints on the model and can be employed in cases where other algorithms fail.

Methods based on random truncation

The ProPPA framework includes implementations of the two algorithms presented in the previous chapter. The random truncation-based methods complement ABC as tools for handling systems with infinite state-spaces; in contrast with ABC, however, they are asymptotically exact methods.

Inference using fluid approximation

The final two methods are based on continuous approximations of the model. Although generally applicable, they are expected to be more accurate as the populations of the species in the model increase in size. The first method considers a deterministic approximation of the stochastic dynamics, greatly simplifying inference. We have not given a formal definition of such a *fluid* or *mean-field* semantics for ProPPA, although this has been done for similar languages like PEPA [127]. The algorithm constructs an ODE for each species, the solution of which gives the average count for that species at any time point. The likelihood is computed by simply solving the ODEs and comparing with the observations, assuming Gaussian noise. The simplicity of the approach should be considered carefully: while it offers computational savings, it ignores the stochasticity of the original model, which may be unsatisfactory. It is worth pointing out that the ProPPA model itself does not need to undergo any changes; it is simply given a different interpretation in terms of this continuous view. It is straightforward to construct the necessary ODEs automatically from the formal description of the model, something which would involve more effort if working directly with the stochastic process instead of a high-level language.

Inference based on the Linear Noise Approximation

A more elaborate approximation, the LNA includes stochasticity in the resulting continuous state description. The counts of the species are no longer assumed to be deterministic but normally distributed at each time point. ODEs for the mean and covariance of this distribution are constructed and the likelihood is computed following the method of Fearnhead *et al.* [32]. All of the quantities needed to construct and solve the ODEs are easily obtained by the stoichiometry matrix, the compiled rate functions

of the model and their derivatives (automatically calculated based on the syntax of the rate function). However, the need to compute derivatives means that this algorithm is not applicable when the rate expressions in the model involve floor or Heaviside functions. As with the fluid approximation, this inference method is more appropriate for models with high counts of species, although this does not mean it cannot be applied to smaller models as well.

5.1.4 Further analysis

Although the main goal of the current version of the ProPPA tool is performing inference, the framework includes additional functionality for working with model files. These methods can be called on any syntactically correct model, either to analyse it or to interpret and visualise the inference results.

Chief among them is the facility to simulate a model until a given time. The simulation can be performed under either stochastic semantics (using the SSA) or a fluid semantics, where the species counts are treated as continuous quantities. For a model containing uncertain parameters, their value must be fixed for the simulation; this can be done by providing either specific numerical values, or a distribution from which to sample values (with the prior specified in the model being the default choice for the latter case). The simulation results, along with basic statistics, can be plotted directly.

The sequence of parameter samples that are returned by the algorithm are saved to an output file. These results can then be used to, for instance, produce histograms illustrating the posterior distribution, as shown in the next section. They can also form the basis of further analysis: for example, predictions of the system's behaviour, conditioned on the observations, can be made by choosing a value from the posterior samples, simulating the model for this parameter value, and repeating this process.

5.1.5 Implementation details

The ProPPA framework is implemented in Python 3 and makes use of various libraries for efficiency and convenience. The parser was based on the `pepapot` tool developed by Allan Clark to support analysis of PEPA and Bio-PEPA models, itself using the `pyparsing` library [86], and was extended to support the novel features of ProPPA. Numerical computations are performed through the widely-used `NumPy` and `SciPy` packages [128, 63], while plotting uses the `matplotlib` library [60].

5.2 An epidemiological case study

This section describes our use of ProPPA in an epidemiological model for the spreading of mumps. We have adapted an existing model and used real data to infer the parameters governing the various processes involved. Our goals were to examine the applicability of the language and modelling framework on a larger model with real-world data, and to gain insight into the particular system which could potentially be of medical use.

5.2.1 Background

Mumps is a viral disease that used to commonly afflict children. There is no known cure, but the infection passes naturally. While symptoms are generally mild (and one in three patients have no noticeable symptoms), more serious complications can arise, especially in adult patients. Inoculation of children is now common by means of the combined MMR (measles, mumps and rubella) vaccine. Although it was introduced in the 1980s, allegations that surfaced in 1998 (and were later disproven) linking it to autism discouraged many parents from vaccinating their children during that time [97]. An important aspect of mumps infection that will affect the modelling is its seasonal behaviour: infection is much more common during certain months of the year than others.

5.2.2 Model

The model we used has been created by Dalila Hamami and Carron Shankland at the University of Stirling to analyse existing data for mumps infections and make future predictions. It is a more sophisticated variant of the SIR model, with two critical alterations. Firstly, it accounts for both the immediate impact of vaccinations and the lapsing of their effectiveness. This is achieved by explicitly representing individuals who have been immunised as distinct types of agents. There are two such classes, MMR1 and MMR2, corresponding to one or two doses of immunisation respectively.

The second change is that infection dynamics are time-dependent, reflecting the seasonality of the infectious behaviour of the virus. Although the ProPPA definition does not support references to time in the model description, preliminary analysis indicated that eliminating the seasonality (i.e. making the dynamics time-invariant) does not match the oscillating patterns of behaviour observed. The way in which the time dependence was included in the model and its implications are detailed in subsequent

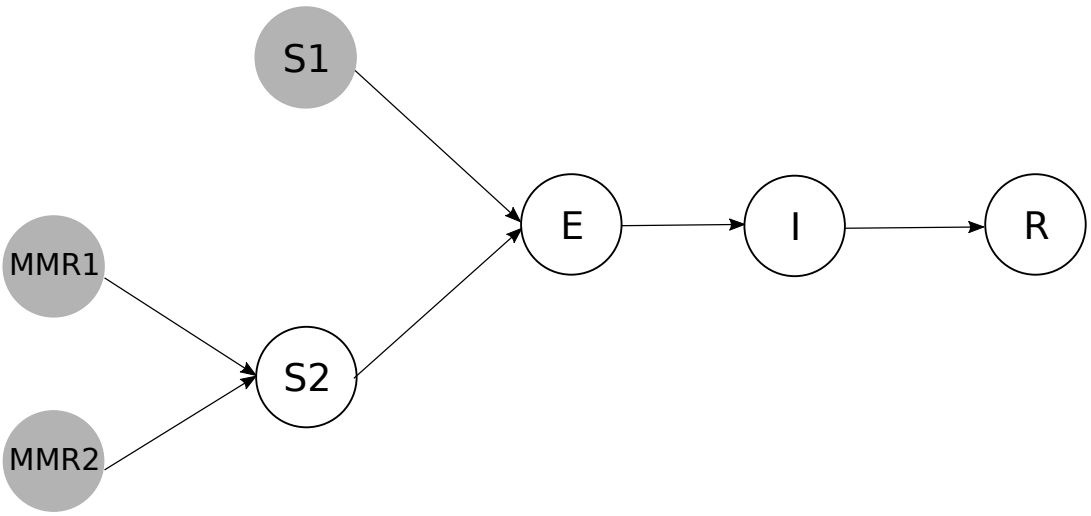


Figure 5.3: States of an individual in the mumps infection model used in the case study. Possible states that individuals can be born into are shown in grey. Death is possible from all states.

sections.

More specifically, there are seven species in the model, each representing a possible state of an individual (Figure 5.3). $S1$ refers to susceptible, non-vaccinated individuals. $MMR1$ and $MMR2$ are individuals that have been vaccinated with one or two dosages, respectively. The protection of the vaccine can be lost with time, making individuals susceptible again ($S2$). A susceptible individual can contract the disease; this change is reflected in them moving to the exposed (E) state. Once symptoms are displayed, they are classified as infected (I). After the disease has passed, recovered individuals (R) are immune to further infection.

The total population does not remain constant. New individuals are "born" into S, $MMR1$ and $MMR2$; the corresponding rates reflect the birth rate, combined with the rate of vaccination for the latter two. Individuals can die at any state. Since the death rate from mumps infections is very small, we assume that the death rate of individuals is not affected by infection, and the parameter controlling this is therefore shared across all states. Finally, the model includes a pool of infected individuals that are initially located outside the system and slowly enter it, to represent infections originating from outside the initial population (such as by people moving into the region).

Name	Description	min	max
β	Low transmission rate	0.1	0.5
β_1	High transmission rate (unvaccinated individuals)	0.6	1.6
β_2	High transmission rate (vaccinated individuals)	0.6	1.6
α	Rate of moving from infected to exposed	0.04	0.085
γ	Recovery rate	0.11	0.17
imrate	Immigration rate	0.2	0.7
Δ	Rate of waning of immunity	0.00014	0.00028

Table 5.2: Uncertain parameters of the mumps model.

5.2.3 ProPPA model

The original model was written in Bio-PEPA, which uses a similar syntax to ProPPA, but where parameters have fixed values. Due to the similarities, writing the ProPPA model was straightforward; the changes involved changing some of the parameters to uncertain, and rewriting parts of the model that depended on Bio-PEPA features not available in ProPPA. The resulting model has seven uncertain parameters, summarised in Table 5.2. Ranges for their values were provided by Dalila Hamami and come from expert knowledge. Uniform priors were used for all uncertain parameters, and the initial state was based on demographic data.

The main change, as indicated before, was the treatment of time. ProPPA does not support time-dependent rates, neither in its semantics nor in the implementation, but in this case the inhomogeneity is an essential part of the model. Using the fluid approximation provides an easy solution to this problem: we can use an additional species T to represent time, giving it an initial count of 0, and a reaction which increases its count at a constant rate of 1 (in a sense, representing a clock ticking). As the species has no other interactions, this ensures that its value always matches the current time. With this work-around, reaction rates can now refer to the value of T to simulate dependence on time.

The ProPPA model is shown in Figures 5.4 and 5.5 (the code is written in one file but split here for better presentation). In addition to this, two other files are needed. The first is the observations file `mumps_obs`, containing the data used for inference. The second, `mumps_config`, specifies parameters for the inference algorithm used (Section 5.2.5).

```
D_R = 0.000037; // death rate
Beta1 = Uniform(0.6,1.6); //high season
Beta2 = Uniform(0.6,1.6); //high season
Beta = Uniform(0.1,0.5); //low season
v2= 0.0000028; //"birth" rate for vaccines at ages 0-2
v3= 0.000025; //"birth" rate for vaccines at ages aged 3-5
v1 = 0.0000021; //"birth" rate for unvaccinated
Alpha = Uniform(0.04,0.085);
Gama = Uniform(0.11,0.17); // recovery rate
imrate1 = Uniform(0.2,0.7); // immigration
Delta = Uniform(0.00014,0.00028); // waning immunity rate

kineticLawOf BIRTH1:  v1 * N  ;
kineticLawOf BIRTH2:  v2 * N;
kineticLawOf BIRTH3:  v3 * N ;
kineticLawOf  MMR1_S2: MMR1 * 2 * Delta;
kineticLawOf  MMR2_S2: MMR2 * Delta;
kineticLawOf Death_MMR1 : D_R * MMR1;
kineticLawOf Death_MMR2 : D_R * MMR2;
kineticLawOf immigration : imrate1/10000 * O;
kineticLawOf  S1_E: (Beta1 * (1-H((month-9)*(4-month))) + Beta * (H((month-9)*(4-month)))) *
               S1 * I / N ;
kineticLawOf  S2_E: (Beta2 * (1-H((month-9)*(4-month))) + Beta * (H((month-9)*(4-month)))) *
               S2 * I / N ;
kineticLawOf  E_I: Alpha *  E;
kineticLawOf  I_R: Gama *  I;
kineticLawOf Death_S1 : D_R * S1;
kineticLawOf Death_I : D_R * I ;
kineticLawOf Death_E : D_R * E;
kineticLawOf Death_S2 : D_R * S2;
kineticLawOf Death_R : D_R * R;
kineticLawOf clock : 1;
```

Figure 5.4: ProPPA mumps model: parameters and kinetic laws. `N` and `month` are abbreviations used here for simplification: `N = S1 + E + I + R + S2 + MMR1 + MMR2` is the total population at any time, and `month = floor(T/30) - 12*floor(floor(T/30)/12)` calculates the month of the year, ranging from 0 to 11.

```
S1 =  (BIRTH1,1) >>  S1  + (S1_E,1) << S1 + (Death_S1,1) << S1 ;
S2 =  (S2_E,1) << S2 + Death_S2 << S2 + (MMR2_S2,1) >> S2 +(MMR1_S2,1) >> S2;
E = (S1_E,1) >> E +(S2_E,1) >> E +(E_I,1) << E+  (Death_E,1) << E  ;
I = (E_I,1) >> I +(I_R,1) << I +   Death_I << I + immigration>>I + (S1_E,1) (.) I
    + (S2_E,1) (.) I;
R = (I_R,1) >> R+ (Death_R,1) << R ;
O = immigration <<;
MMR1 = (BIRTH2,1) >>  MMR1  + (MMR1_S2,1) << MMR1+ (Death_MMR1,1) << ;
MMR2 = (BIRTH3,1) >> MMR2 + (MMR2_S2,1) << MMR2 + (Death_MMR2,1) <<  ;
T = clock >> ;

S1[1000000]<*> S2[0]<*>  E[0]<*> I[20]<*> R[3518600]<*> MMR1[273541]
      <*>MMR2[250000] <*> O[10000] <*> T[0]

observe(mumps_obs);
infer(fluid);
configure(mumps_config);
```

Figure 5.5: ProPPA mumps model: species definitions, initial state and configuration.

5.2.4 Data

For the purposes of this study, we used data supplied by Health Protection Scotland, a division of the Scottish National Health Service. The data consisted of all reported instances of mumps between 2004 and 2014 across all of Scotland. The data was anonymised and the only information extracted was the number of reported instances each day. From this, we estimated the actual number of infected individuals: denoting by r_t the reported number of incidents on day t, the estimated number of infected individuals on that day is

$$I_t = \sum_{i=0}^{6} r_{t+i}$$

In other words, we consider that an infected individual will be reported some time within a seven-day window from the day of infection.

One aspect hampering the analysis of the data is the unknown but potentially large scale of under-reporting. Previous studies in different countries [122] have examined the rate of this phenomenon for other diseases, and indicate that mumps is also affected by it [71] — however, no concrete rate appears to have been reported in the latter case. A previous case study of computational modelling of measles [11] addressed this issue by rescaling the data to match the expected behaviour of the system. We adopted a

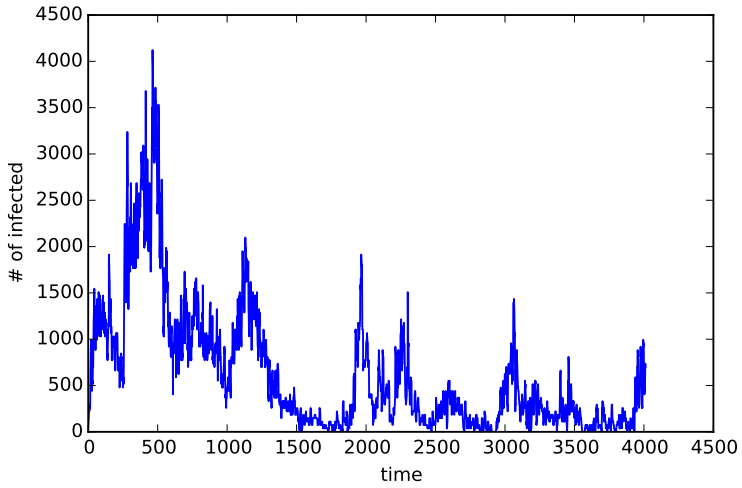


Figure 5.6: Observations used in the case study: estimated infected individuals vs time (in days)

similar approach, resulting in the data shown in Figure 5.6.

Two important features of the data are immediately apparent. Firstly, there is a big spike in the number of infections about 18 months into the period covered by the data. It is not known whether this behaviour can be explained fully by the model or if it is the result of factors not captured therein, such as the drop in the number of vaccinations in the past. As such, it is interesting to see how much of an impact the presence of this spike has on the inference results. Secondly, there is a general oscillatory trend in the number of infected individuals. This will be part of our evaluation of the inference results, as we would like a fitted model to at least qualitatively capture this pattern.

5.2.5 Experiments using a deterministic approximation

The large populations in the model justify the use of continuous approximations to analyse it. We first used the fluid sampler with parameters shown in Table 5.3 (specified through the configuration file).

We started by only using the second part of the observations (from time 2000 onward); the sampling results for each parameter are shown in Figure 5.7. Interestingly, for most parameters the samples are heavily distributed towards one extreme of the prior range. This may indicate a number of things. First, the ranges may be misspecified: although the values were provided by experts, some are based on estimates rather than documented behaviour. Assuming that the ranges are correct, this could suggest that

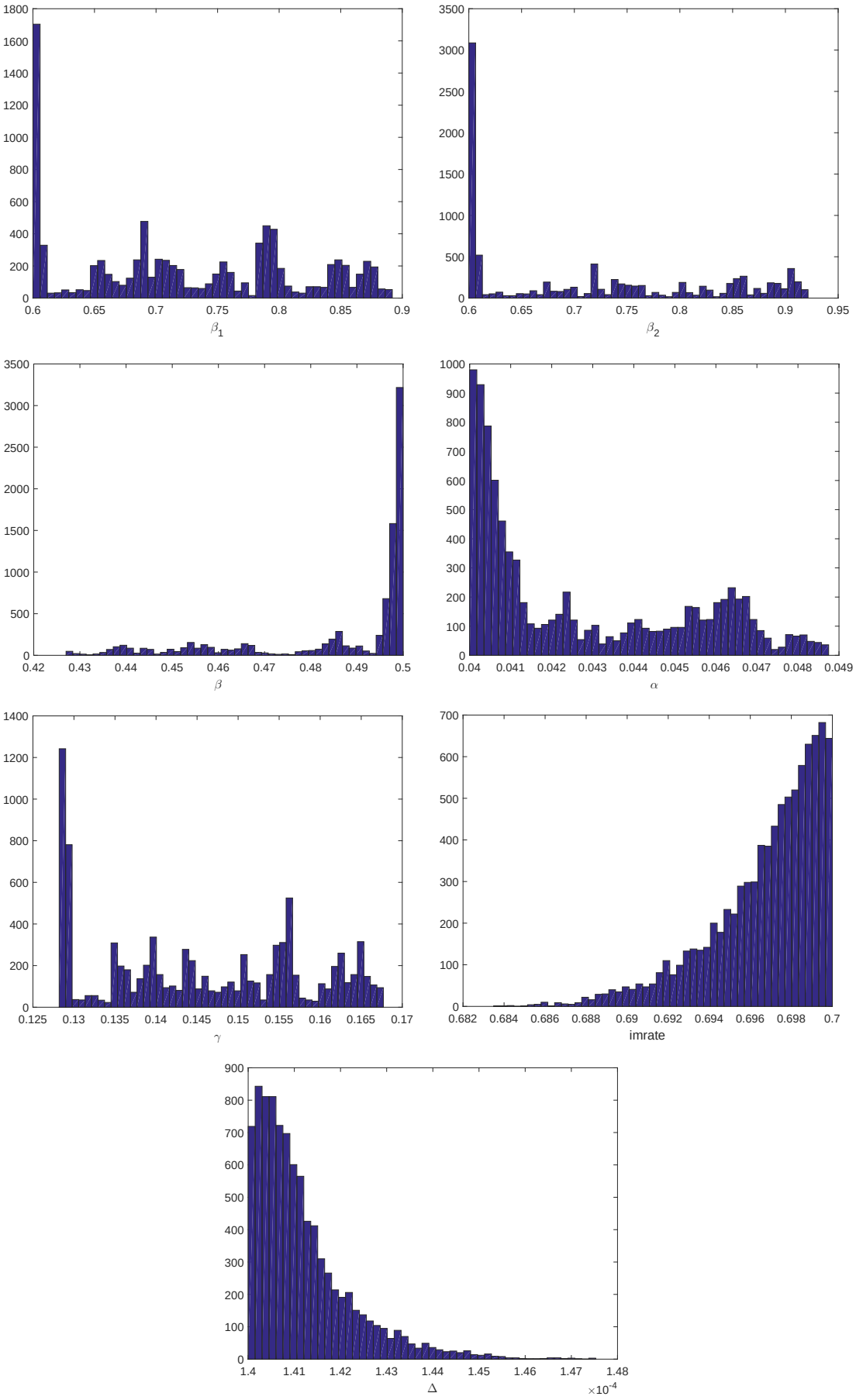


Figure 5.7: Histograms of posterior samples for the parameters of the mumps model using observations from time 2000 onwards.

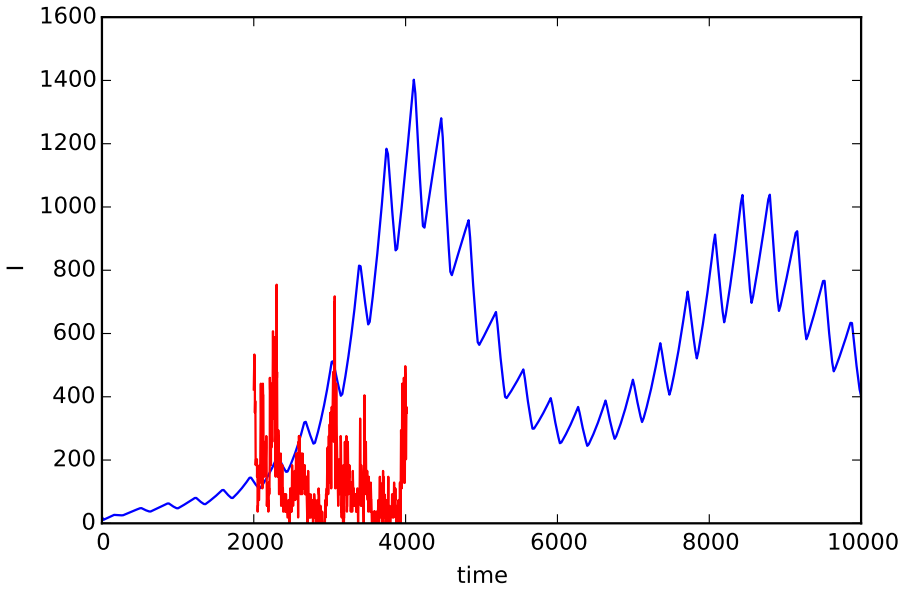
Parameter	Value
Noise variance	10000
β_1 proposal std	0.001
β_2 proposal std	0.001
β proposal std	0.0005
α proposal std	0.0001
γ proposal std	0.0001
imrate proposal std	0.001
Δ proposal std	0.0000005
Number of samples	10000

Table 5.3: Configuration of the fluid sampler, including the standard deviation (std) of the proposal distribution used for each parameter.

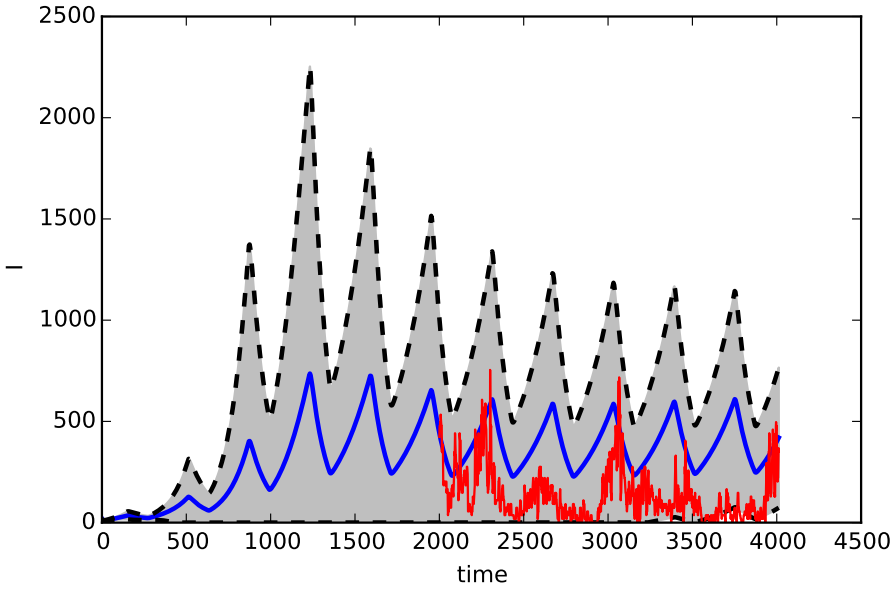
aspects of the model or the preprocessing of the data do not reflect the true situation.

To check how well the sampling output matches the data, we simulated the model using a parameterisation with high posterior probability according to the results (Figure 5.8a). While the fit is not entirely satisfactory, note that the simulation does produce sustained oscillations, even past the end of the observations. This also appears to be the case if we consider multiple parameter values and take the average prediction (Figure 5.8b); in fact, the average seems to follow the observations more closely. In contrast, the average behaviour of the model according to the priors is significantly different (Figure 5.9).

We repeated the experiment, now considering the full range of observations: again, the output was oscillatory, both for single parameters (Figure 5.10a) and on average (Figure 5.10b). The results appear much closer to the observations in this case, even if they don't fully capture the amplitude of the first spike. Perhaps surprisingly, it seems that the presence of the initial spike in the observed data improves the fit even on the second half of the time interval. It is still not clear whether the spike can be accurately reproduced by the model in its current form, but these results may indicate that it is not wholly external to the model.



(a)



(b)

Figure 5.8: Posterior simulation results when using observations from time 2000 onwards (shown in red): (a) Output for the model for a single parameterisation ($\beta_1 = 0.6, \beta_2 = 0.6, \beta = 0.5, \alpha = 0.04, \gamma = 0.13, imrate = 0.7, \Delta = 0.00014$); (b) Average output over 1000 parameter values drawn from the posterior (mean $\pm$ 1 standard deviation).

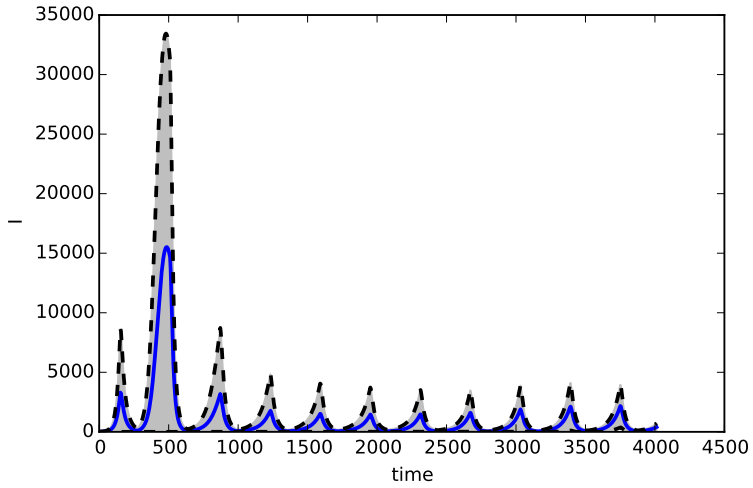


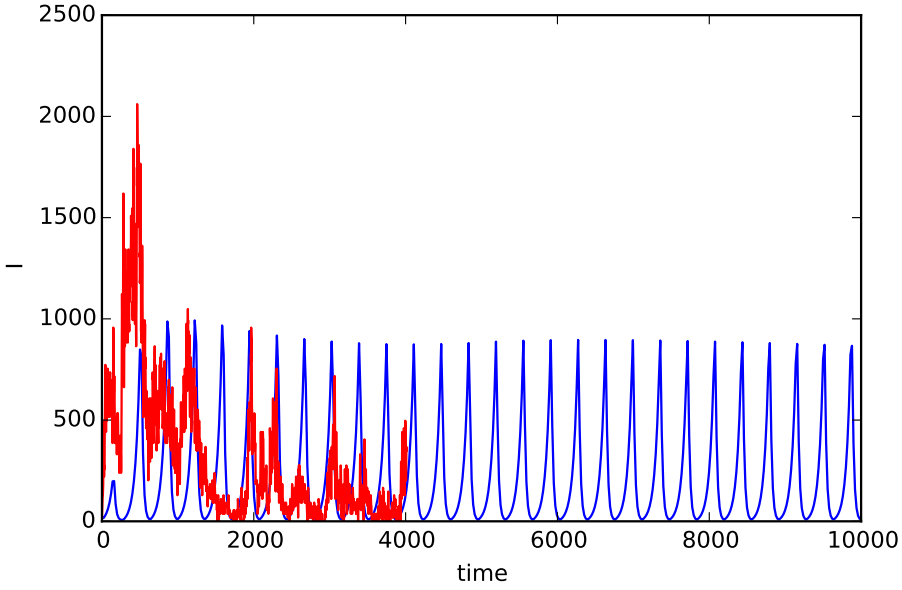
Figure 5.9: Simulated output using 2000 parameter values drawn from the prior (mean $\pm$ 1 standard deviation).

5.2.6 On other continuous approximations

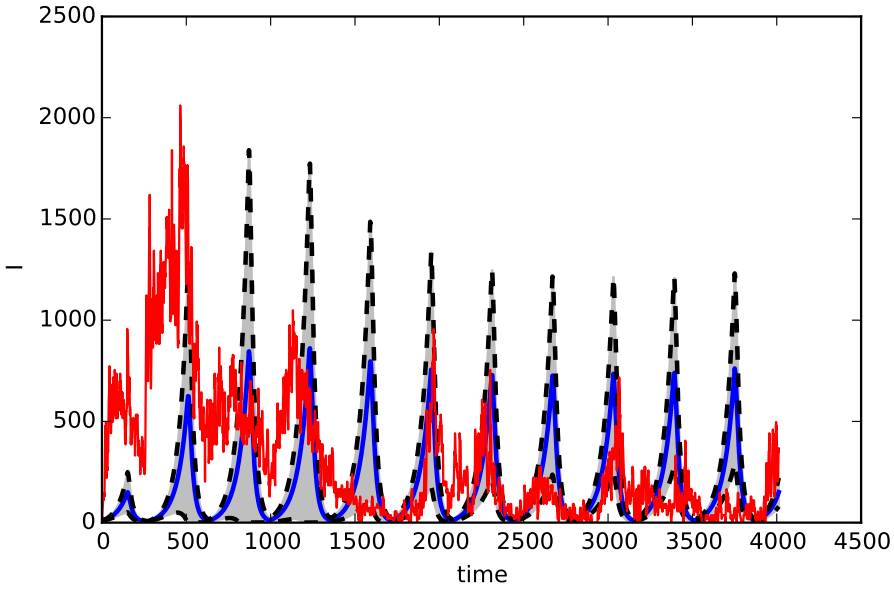
It would be interesting to see how the results change under different algorithms. The LNA-based sampler may seem to be a good candidate for further experimentation at first glance, in order to maintain some of the stochasticity of the model. However, it is problematic to use in this case, due to a number of factors.

The first, as indicated previously, has to do with the presence of explicit dependence on time in the rates. While this was easy to deal with when working with the deterministic approximation, the same solution cannot be applied here. Under the LNA, all the species have an uncertainty associated with their level. This defeats the point of encoding time (whose values are deterministic) as a species. To counter this issue, which may arise in other models of interest, we developed a variant of the framework which allows rate expressions to refer to time, appropriately modifying other parts of the framework in turn. Note, however, that the ProPPA language as theoretically defined in Chapter 3 cannot capture time-inhomogeneous behaviour, and we have not attempted to introduce that aspect into the semantics — although an appropriate extension may not be difficult to formalise.

Another complicating factor arises from the use of discontinuous functions in the model — specifically, the floor and Heaviside functions. These are used to control the change in transmission rate by calculating the month of the year from the time. The LNA sampler requires computing the derivatives of the rate functions with respect to



(a)



(b)

Figure 5.10: Posterior simulation results when using all the observations (shown in red): (a) Output for the model for a single parameterisation ($\beta_1 = 1.15, \beta_2 = 0.6, \beta = 0.1, \alpha = 0.046, \gamma = 0.13, imrate = 0.7, \Delta = 0.00014$); (b) Average output over 1000 parameter values drawn from the posterior (mean $\pm$ 1 standard deviation).

species levels; as such, these discontinuities make it inapplicable. While it is possible to approximate the Heaviside function with a sharply-increasing sigmoid curve, we could not find a good approximation for the floor function, and its removal significantly altered the behaviour of the model.

Finally, there may be practical difficulties due to only observing one out of the seven species in the model. The likelihood computation involves calculating variances for all species, which are updated according to the observations. The low dimension of the latter might create numerical instabilities during the solution; however, due to the above issues, it was not possible to verify whether this is indeed the case or whether the low number of observables does not pose any problems.

5.2.7 Discussion

We have shown how the ProPPA framework can be applied to learn the parameters of a model from real-world data. A few things are worth noting from the above experience. "Translating" the model into ProPPA, although not entirely trivial, was relatively straightforward. This is encouraging for the use of ProPPA with models written in other languages of a similar style, and it suggests that the framework could be a useful tool for performing parameter estimation in a variety of existing models.

The main challenges had to do with how to capture features that are not part of ProPPA, such as time-dependence. Of the samplers available in the core inference engine, the ones based on continuous approximations were most appropriate given the high populations involved. Unfortunately the particular details of this model meant that only one sampler could be used. We believe that such small difficulties or limitations are not an obstacle, especially considering that ProPPA allows easy access to automated, statistically rigorous estimation algorithms. This is an attractive alternative to the current effective *status quo*: manual tuning of the parameters, as had happened with the model in question, which can be a time-consuming process with uncertain outcomes.

Chapter 6

Conclusions

6.1 Summary

This thesis has presented an integrated framework for specification and inference of population-based stochastic systems with uncertain parameters. Our work was motivated by observations regarding the state of the art in describing stochastic systems: current high-level languages assume a full knowledge of the system, seldom found in realistic applications, while machine learning techniques are applicable to low-level mathematical descriptions of the underlying process. We proposed ProPPA, a new language which embeds features of the probabilistic programming paradigm into a formal process algebra, allowing the modeller to include uncertain knowledge and observations, and perform inference automatically. The existence of uncertainty in the language meant that CTMCs were not a sufficient mathematical object to base its semantics upon. This led us to define Probabilistic Constraints Markov Chains (pCMCs), by adapting the previously defined Constraint Markov Chains to the continuous-time domain and shifting from a qualitative to a quantified expression of uncertainty.

In one sense, the resulting framework provides a more sophisticated, statistically rigorous alternative to existing parameter learning techniques for process algebra models while remaining automatic and without requiring expert machine learning knowledge on the part of the user. Perhaps its more important contribution, however, is in displaying how machine learning approaches can be integrated with formal modelling, and how ideas from probabilistic programming can be applied to high-level systems modelling languages. In introducing ProPPA, we address an identified gap in the treatment of continuous-time systems with probabilistic programming techniques.

During the development of this framework, it became clear that existing methods for

Bayesian inference of CTMCs were lacking when infinite state-spaces were encountered. Therefore, as part of this effort, we developed two new algorithms by making use of the random truncations approach, long known in the physics community and recently introduced in the machine learning literature. The resulting methods compare favourably with state-of-the-art algorithms and offer an alternative to conservative truncation strategies such as the FSP, while retaining statistical guarantees; the latter point sets them apart from heuristic-based methods like ABC, the performance of which also suffers from severe sensitivity to the parameters chosen for the algorithm.

ProPPA is intended to be a platform both for performing inference and analysing the behaviour of uncertain stochastic systems. The full framework includes a number of algorithms in its core inference engine, which can be chosen according to the user's requirements and offer different interpretations of the underlying dynamics. It also provides various methods for further simulation and analysis of a model and the inference results, to support it as a free-standing platform. Throughout the thesis, we have given examples of small ProPPA models used for testing the algorithms. A large-scale model of disease-spreading served as a case study; the language was shown to be expressive enough to capture the dynamics, and the additional functionality included in the framework made it easy to analyse the results.

6.2 Future work

This section outlines how our work can serve as a basis for further development in various directions. We believe that pCMCs can be a useful formalism for describing stochastic systems with uncertain or incomplete specification. We first give some basic ideas for defining equivalence relations in this context, before presenting further ideas for extending the work contained in the thesis.

6.2.1 Equivalences for pCMCs

A common question in formal analysis of systems is to examine when two states or components are interchangeable. One of the benefits of this kind of analysis is that it allows one to construct reduced models of a system which exhibit the same behaviour. The question is generally framed in the definition of an equivalence relation or, equivalently, a partitioning of the state-space, which must fulfil appropriate conditions. Any states that belong to the same partition, i.e. are related through the equivalence relation, are

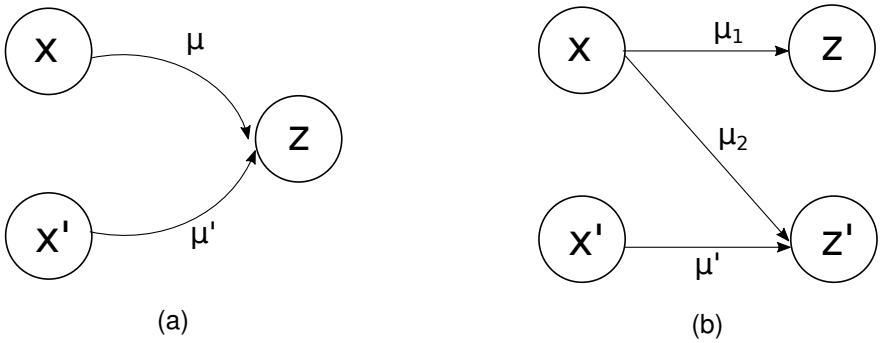


Figure 6.1: Example models for demonstrating equivalences for pCMCs.

then considered to behave identically. The particular conditions required depend on the dynamics of the system and the meaning attributed to the idea of "identical" behaviour.

In this section, we give a preliminary treatment of this problem in the setting of pCMCs. Our approach is similar to notions of equivalence defined for PEPA [58] and Bio-PEPA [35], which are in turn inspired by the *probabilistic bisimulation* of Larsen and Skou [75]. Intuitively, two states s, s' of a CTMC are bisimilar if the total rate of going from s to an equivalence class is the same as from s' to that class, for all equivalence classes[1].

In order to be applicable in the pCMC setting, this idea must be adapted to account for uncertainty in the transition rates. The examples in Figure 6.1 illustrate simple cases of states which could be considered equivalent. First consider the case where two states x, x' have only one outgoing transition each, with the same target. It is natural to assume that x and x' are equivalent if the rate of that transition follows the same distribution, i.e. if $\mu = \mu'$. The case of Figure 6.1b is slightly more complex. If we assume that z and z' are equivalent, then, for x and x' to be equivalent, we want the *total* rate of transitioning from x to z or z' to be the same as the rate of going from x' to z'.

We now generalise and formalise this to propose a definition of equivalence for pCMCs. For brevity of notation, we rewrite the constraint function ϕ of a pCMC as a labelled transition relation $\mathcal{R}_\phi$, such that

$$(s, \mu, s') \in \mathcal{R}_\phi \text{ iff } \phi(s, s', r) = \mu(r) \text{ for all } r$$

This is simply a reformulation to shorten the following definitions. This transition relation, like the constraint function, gives us the distribution of the rate of going from

[1]The definitions of bisimilarity in the cited works also take into account the action by which the transition occurs. We do not make this distinction here because of the way pCMCs are defined and for simplicity, but it can be accommodated by simple modifications to the pCMC definition.

one state to another. We can extract this as:

$$rate\_dist(s,s') = \mu \text{ , where } (s,\mu,s') \in \mathcal{R}_\phi$$

Keeping in mind Lemma 2 in Section 3.4.3 for the distribution of the sum of random variables, this can be extended to retrieve the rate of going from s to a set of states S' :

$$rate\_dist(s,S') = \bigotimes_{s' \in S'} rate\_dist(s,s')$$

where $\bigotimes$ denotes convolution. We can now define the following:

Definition 9. An equivalence relation $\mathcal{R} \subset \mathcal{S} \times \mathcal{S}$ is a *bisimulation* for a pCMC with state-space $\mathcal{S}$ if, for every s, s' with $s\mathcal{R}s'$, it holds that

$$rate\_dist(s,C) = rate\_dist(s',C)$$

for all equivalences classes C of $\mathcal{R}$.

In the style of previous work, this can serve as the basis for further definitions; for instance:

Definition 10. Two states s and s' are *bisimilar* if $s\mathcal{R}s'$ for some bisimulation $\mathcal{R}$.

Our goal here is to give a brief flavour of what notions of equivalence can be considered, and we will not go to more depth.

Consider again the example in Figure 6.1b, and the partition $R = \{\{x,x'\},\{z,z'\}\}$. Assume that the distributions on the transition rates are all Gamma, with

$$\begin{aligned} \mu_1 &\sim Gamma(a_1,b) \\ \mu_2 &\sim Gamma(a_2,b) \\ \text{and } \mu' &\sim Gamma(a',b) \end{aligned}$$

Considering the distribution of the sum of two independent Gamma-distributed random variables with the same rate, the total rate of transitioning from x to $\{z,z'\}$ is $Gamma(a_1+a_2)$. It then follows that R is a bisimilarity if and only if $a' = a_1 + a_2$.

The above definition intuitively says that the rates of transitions from equivalent states follow the same distributions. This is equivalent to asking that the corresponding transition *times* have the same distribution. This can be seen by starting with the

cumulative distribution function (cdf) of the transition time, assuming that the rate is known to be r: $P(t \leq t_0 \mid r) = 1 - e^{-rt_0}$. By marginalising out the rate, we have

$$\begin{aligned} P(t \leq t_0) &= \int \left(1 - e^{-rt_0}\right) \mu(r) \mathrm{d}r \\ &= \int \mu(r) \mathrm{d}r - \int e^{-rt_0} \mu(r) \mathrm{d}r \\ &= 1 - \mathbb{E}_{\mu}\left[e^{-rt_0}\right] \end{aligned}$$

Now consider another distribution $\mu'(r)$ on the rate. In order for the cdf of the times to match, we must have $\mathbb{E}_{\mu}[e^{-rt_0}] = \mathbb{E}_{\mu'}[e^{-rt_0}]$ for all t_0. In other words, the distributions μ and μ' must have the same moment-generating functions, and must therefore be the same.

The definition given here may be too strict, as it requires exact equality of distributions. It may be interesting, therefore, to consider approximate equivalence relations, based on a notion of distance of distributions. There is a significant body of work on approximate equivalences, originating in both the formal modelling and the control theory communities, and the references given here are only indicative. However, attention has been focused on discrete-time systems [126], as well as continuous-state [27] or hybrid systems [65] (ones with both discrete and continuous state variables); the question does not appear to have been examined for systems with discrete state and continuous time, like CTMCs. The idea of distances between distributions has been explored in this setting to derive distances between states [20], but not approximate equivalence relations. Another prospect is defining simulation relations, expressing the concept of one state being "faster" than another, which could then be related to the concept of stochastic orders [93]. There is undoubtedly room for further elaboration on the subject.

6.2.2 Other directions

In addition to the above idea, the work presented here can be extended in other paths. Regarding the inference algorithms, while the proposed methods perform better than the state-of-the-art, considering other existing techniques for analysis of CTMCs may still offer benefits at different points. One such instance concerns the Gibbs (or Gibbs-like) samplers of Section 4.4. As discussed in that chapter, one of the reasons why performance decreases with larger population sizes is that the reaction rates become very high, resulting in a very fine time-discretisation and a corresponding long chain for the FFBS algorithm. This is a known weakness of uniformisation, namely, its bad

performance when a broad range of reaction rates coexist in the system. Alternative methods such as Adaptive [129] and Fast Adaptive Uniformisation [85] have been proposed for more efficient computation of transient probabilities by using a more flexible time discretisation and dynamically adapting the state-space. It is possible that these methods or similar ideas could be used to also improve the performance of the proposed samplers.

Another point concerns the M-H Roulette sampler of Section 4.3.3 and, in particular, the calculation of successive terms in Equation (4.3). Recall that each term is decomposed into two transient probabilities according to Equation (4.4), with $f^{(N)}$ being used in the calculation of both $p^{(N)}$ and $p^{(N+1)}$. This repetition affords some computational savings; however, each $f^{(N)}$ is recomputed "from scratch". It would be interesting to see whether the computation of the probabilities for a state-space $\mathcal{S}_N$ could be informed by already having performed the computation within $\mathcal{S}_{N-1}$, thus yielding even more significant improvements. It is not trivial to compute the "additional" transient probability for the extended state-space, but one potentially useful way of looking at the problem is to frame it as a model-checking task: essentially, we want to compute the probability of reaching a target state from a known distribution of starting states, while visiting at least one of the "border" states $\vartheta\mathcal{S}_N = \mathcal{S}_N - \mathcal{S}_{N-1}$ and avoiding any states outside $\mathcal{S}_N$. The probability of the trajectories that satisfy this requirement corresponds to the term $p^{(N)}$.

With regards to the ProPPA framework, we are interested in extending it with additional features such as an editor, to provide a more complete and integrated modelling environment. The design of the system intentionally allows for different solvers to be easily added, which gives the option of including algorithms based on, for instance, moment closure approximations. Furthermore, we would like to add support for observations encoded as logic specifications to the framework, as explored in Chapter 3, to complement the algorithms working with time-series. Additional features could promote both the language and the framework as an experimental platform for use in different fields.

The above thoughts are indicative directions in which this work could be extended. More generally, we believe it can potentially serve as a blueprint for further integration between the fields of formal modelling and machine learning. We believe there is a significant potential for work in the intersection of the two approaches, with great benefits to be earned by applying methods from one field to problems of the other. For instance, the Russian Roulette truncation scheme could be applied to problems

such as calculation of transient probabilities via uniformisation. There could also be scope to study and reason about statistical properties of the posterior distribution, which would allow us to give a formal characterisation of the inference process beyond the simple pattern proposed in Section 3.5.2. Another interesting possibility would be to use the results of formal analysis to inform the selection of an appropriate inference algorithm, or to tackle the more complicated problem of structure learning. This thesis has identified some ways in which such integration can occur, opening the possibility of further exploration in this direction.

Bibliography

[1] Agapiou, S., Roberts, G.O., Vollmer, S.J.: Unbiased Monte Carlo: posterior estimation for intractable/infinite-dimensional models. arXiv preprint arXiv:1411.7713 (2014)

[2] Al-Mohy, A.H., Higham, N.J.: Computing the Action of the Matrix Exponential, with an Application to Exponential Integrators. SIAM Journal on Scientific Computing **33**(2) (2011) 488–511

[3] Alur, R., Feder, T., Henzinger, T.A.: The Benefits of Relaxing Punctuality. Journal of the ACM **43**(1) (January 1996) 116–146

[4] Amrein, M., Künsch, H.R.: Rate estimation in partially observed Markov jump processes with measurement errors. Statistics and Computing **22**(2) (2011) 513–526

[5] Anderson, R.M., May, R.M.: Infectious diseases of humans. Volume 1. Oxford University Press (1991)

[6] Andreychenko, A., Mikeev, L., Spieler, D., Wolf, V.: Parameter Identification for Markov Models of Biochemical Reactions. In Gopalakrishnan, G., Qadeer, S., eds.: Computer Aided Verification. Volume 6806 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2011) 83–98

[7] Andrieu, C., Doucet, A., Holenstein, R.: Particle Markov chain Monte Carlo methods. Journal of the Royal Statistical Society: Series B **72**(3) (2010) 269–342

[8] Andrieu, C., Roberts, G.O.: The pseudo-marginal approach for efficient Monte Carlo computations. The Annals of Statistics (2009) 697–725

[9] Aziz, A., Singhal, V., Balarin, F., Brayton, R.K., Sangiovanni-Vincentelli, A.L.: It usually works: The temporal logic of stochastic systems. In Wolper, P., ed.: Computer Aided Verification. Volume 939 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (1995) 155–165

[10] Beaumont, M.A.: Estimation of Population Growth or Decline in Genetically Monitored Populations. Genetics **164**(3) (2003) 1139–1160

[11] Benkirane, S., Norman, R., Scott, E., Shankland, C. In: Measles Epidemics and PEPA: An Exploration of Historic Disease Dynamics Using Process Algebra. Springer Berlin Heidelberg (2012) 101–115

[12] Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)

[13] Bogomolov, S., Henzinger, T.A., Podelski, A., Ruess, J., Schilling, C.: Adaptive Moment Closure for Parameter Inference of Biochemical Reaction Networks. In: Computational Methods in Systems Biology: 13th International Conference, CMSB 2015, Nantes, France, September 16-18, 2015, Proceedings. Springer International Publishing (2015) 77–89

[14] Borgström, J., Gordon, A.D., Greenberg, M., Margetson, J., Van Gael, J.: Measure transformer semantics for Bayesian machine learning. In: Proceedings of the 20th European conference on Programming languages and systems: part of the joint European conferences on theory and practice of software. ESOP'11/ETAPS'11, Berlin, Heidelberg, Springer-Verlag (2011) 77–96

[15] Bortolussi, L., Gast, N.: Mean Field Approximation of Uncertain Stochastic Models. In: 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2016). (2016)

[16] Bortolussi, L., Sanguinetti, G.: Learning and Designing Stochastic Processes from Logical Constraints. In Joshi, K., Siegle, M., Stoelinga, M., D'Argenio, P.R., eds.: Quantitative Evaluation of Systems. Volume 8054 of Lecture Notes in Computer Science. (2013) 89–105

[17] Boys, R., Wilkinson, D., Kirkwood, T.: Bayesian inference for a discretely observed stochastic kinetic model. Statistics and Computing **18** (2008) 125–135

[18] Caillaud, B., Delahaye, B., Larsen, K.G., Legay, A., Pedersen, M.L., Wasowski, A.: Constraint Markov Chains. Theoretical Computer Science **412**(34) (2011) 4373–4404

[19] Capistrán, M.A., Christen, J.A., Velasco-Hernández, J.X.: Towards uncertainty quantification and inference in the stochastic SIR epidemic model. Mathematical Biosciences **240**(2) (2012) 250–259

[20] Cardelli, L., Mardare, R.: The Measurable Space of Stochastic Processes. In: Seventh International Conference on the Quantitative Evaluation of Systems (QEST). (Sept 2010) 171–180

[21] Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M.A., Guo, J., Li, P., Riddell, A.: Stan: A probabilistic programming language. Journal of Statistical Software (2016)

[22] Chattopadhyay, I., Kuchina, A., Süel, G.M., Lipson, H.: Inverse Gillespie for inferring stochastic reaction mechanisms from intermittent samples. Proceedings of the National Academy of Sciences **110**(32) (2013) 12990–12995

[23] Ciocchetta, F., Hillston, J.: Bio-PEPA: A framework for the modelling and analysis of biological systems. Theoretical Computer Science **410**(33-34) (2009) 3065–3084

[24] Cox, R.T.: Probability, Frequency and Reasonable Expectation. American Journal of Physics **14**(1) (1946) 1–13

[25] Danos, V., Feret, J., Fontana, W., Harmer, R., Krivine, J.: Rule-Based Modelling of Cellular Signalling. In: CONCUR 2007 - Concurrency Theory. Volume 4703 of Lecture Notes in Computer Science. Springer Berlin/Heidelberg (2007) 17–41

[26] De Nicola, R., Latella, D., Loreti, M., Massink, M.: A Uniform Definition of Stochastic Process Calculi. ACM Computing Surveys **46**(1) (October 2013) 5:1–5:35

[27] Desharnais, J., Gupta, V., Jagadeesan, R., Panangaden, P.: Metrics for labelled Markov processes. Theoretical Computer Science **318**(3) (2004) 323–354

[28] Doucet, A., Pitt, M.K., Deligiannidis, G., Kohn, R.: Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. Biometrika (2015)

[29] Elerian, O., Chib, S., Shephard, N.: Likelihood Inference for Discretely Observed Nonlinear Diffusions. Econometrica **69**(4) (2001) 959–993

[30] Engblom, S.: Computing the moments of high dimensional solutions of the master equation. Applied Mathematics and Computation **180**(2) (2006) 498–515

[31] Eraker, B.: MCMC Analysis of Diffusion Models With Application to Finance. Journal of Business & Economic Statistics **19**(2) (2001) 177–191

[32] Fearnhead, P., Giagos, V., Sherlock, C.: Inference for reaction networks using the linear noise approximation. Biometrics **70**(2) (2014) 457–466

[33] Fecher, H., Leucker, M., Wolf, V.: Don't Know in Probabilistic Systems. In Valmari, A., ed.: Model Checking Software. Volume 3925 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2006) 71–88

[34] Fox, B.L., Glynn, P.W.: Computing Poisson Probabilities. Communications of the ACM **31**(4) (April 1988) 440–445

[35] Galpin, V.: Equivalences for a biological process algebra. Theoretical Computer Science **412**(43) (2011) 6058–6082

[36] Gardiner, C.W.: Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences. Springer Series in Synergetics (1985)

[37] Gardner, T.S., Cantor, C.R., Collins, J.J.: Construction of a genetic toggle switch in Escherichia coli. Nature **403**(6767) (2000) 339–342

[38] Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B.: Bayesian Data Analysis. Taylor & Francis (2014)

[39] Gelman, A., Roberts, G.O., Gilks, W.R.: Efficient Metropolis Jumping Rules. Bayesian Statistics **5**(599–608) (1996) 42

[40] Georgoulas, A., Hillston, J., Milios, D., Sanguinetti, G.: Probabilistic Programming Process Algebra. In Norman, G., Sanders, W., eds.: Quantitative Evaluation of Systems. Volume 8657 of Lecture Notes in Computer Science. Springer International Publishing (2014) 249–264

[41] Georgoulas, A., Hillston, J., Sanguinetti, G.: ABC-Fun: A Probabilistic Programming Language for Biology. In Gupta, A., Henzinger, T.A., eds.: Computational Methods in Systems Biology. Volume 8130 of LNCS. (2013) 150–163

[42] Georgoulas, A., Hillston, J., Sanguinetti, G.: Unbiased Bayesian inference for population Markov jump processes via random truncations. Statistics and Computing (2016) 1–12

[43] Geyer, C.J.: Practical Markov Chain Monte Carlo. Statistical Science **7**(4) (11 1992) 473–483

[44] Gibson, M.A., Bruck, J.: Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels. The Journal of Physical Chemistry A **104**(9) (2000) 1876–1889

[45] Gillespie, C.S., Golightly, A.: Bayesian inference for generalized stochastic population growth models with application to aphids. Journal of the Royal Statistical Society: Series C **59**(2) (2010) 341–357

[46] Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. Journal of Physical Chemistry **81**(25) (1977) 2340–2361

[47] Gillespie, D.T.: Approximate accelerated stochastic simulation of chemically reacting systems. The Journal of Chemical Physics **115**(4) (2001) 1716–1733

[48] Golightly, A., Henderson, D.A., Sherlock, C.: Delayed acceptance particle MCMC for exact inference in stochastic kinetic models. Statistics and Computing **25**(5) (2014) 1039–1055

[49] Golightly, A., Wilkinson, D.J.: Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo. Interface Focus (2011)

[50] Goodman, N.D., Mansinghka, V.K., Roy, D.M., Bonawitz, K., Tenenbaum, J.B.: Church: A language for generative models. In: Uncertainty in Artifical Intelligence. (2008) 220–229

[51] Goodman, N.D., Stuhlmüller, A.: The Design and Implementation of Probabilistic Programming Languages (2014) `http://dippl.org`. Accessed: 2016-2-24.

[52] Gordon, A.D., Graepel, T., Rolland, N., Russo, C., Borgstrom, J., Guiver, J.: Tabular: A Schema-driven Probabilistic Programming Language. In: Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. POPL '14, New York, NY, USA, ACM (2014) 321–334

[53] Gordon, A.D., Henzinger, T.A., Nori, A.V., Rajamani, S.K.: Probabilistic Programming. In: Proceedings of the on Future of Software Engineering. FOSE 2014, New York, NY, USA, ACM (2014) 167–181

[54] Gordon, N.J., Salmond, D.J., Smith, A.F.M.: Novel approach to nonlinear/non-gaussian bayesian state estimation. IEEE Proceedings F - Radar and Signal Processing **140**(2) (April 1993) 107–113

[55] Götz, N., Herzog, U., Rettelbach, M.: Multiprocessor and distributed system design: The integration of functional specification and performance analysis using Stochastic Process Algebras. In Donatiello, L., Nelson, R., eds.: Performance Evaluation of Computer and Communication Systems: Joint Tutorial Papers of Performance '93 and Sigmetrics '93. Springer Berlin Heidelberg (1993) 121–146

[56] Green, P.J., Łatuszyński, K., Pereyra, M., Robert, C.P.: Bayesian computation: a summary of the current state, and samples backwards and forwards. Statistics and Computing **25**(4) (2015) 835–862

[57] Hajiaghayi, M., Kirkpatrick, B., Wang, L., Bouchard-Côté, A.: Efficient Continuous-Time Markov Chain Estimation. In: Proceedings of the 31st International Conference on Machine Learning (ICML-14). (2014) 638–646

[58] Hillston, J.: A Compositional Approach to Performance Modelling. Cambridge University Press (1996)

[59] Hoare, C.A.R.: Communicating sequential processes. Springer (1978)

[60] Hunter, J.D.: Matplotlib: A 2D graphics environment. Computing In Science & Engineering **9**(3) (2007) 90–95

[61] Jacob, P.E., Thiery, A.H.: On nonnegative unbiased estimators. Annals of Statistics **43**(2) (04 2015) 769–784

[62] Jensen, A.: Markoff chains as an aid in the study of Markoff processes. Scandinavian Actuarial Journal **1953**(sup1) (1953) 87–91

[63] Jones, E., Oliphant, T., Peterson, P., et al.: SciPy: Open source scientific tools for Python (2001–) `http://www.scipy.org/` [accessed 6 October 2016].

[64] Jonsson, B., Larsen, K.G.: Specification and refinement of probabilistic processes. In: Logic in Computer Science, 1991. LICS '91., Proceedings of Sixth Annual IEEE Symposium on. (1991) 266–277

[65] Julius, A.A., Girard, A., Pappas, G.J.: Approximate bisimulation for a class of stochastic hybrid systems. In: 2006 American Control Conference. (June 2006) 4724–4729

[66] Kaminski, B.L., Katoen, J.P., Matheja, C., Olmedo, F.: Weakest Precondition Reasoning for Expected Run–Times of Probabilistic Programs. In Thiemann, P., ed.: 25th European Symposium on Programming, ESOP 2016. Springer Berlin Heidelberg (2016) 364–389

[67] Katoen, J.P., Klink, D., Leucker, M., Wolf, V.: Three-valued abstraction for probabilistic systems. The Journal of Logic and Algebraic Programming **81**(4) (2012) 356–389 Special Issue: NWPT 2009.

[68] Kazeroonian, A., Hasenauer, J., Theis, F.: Parameter Estimation for Stochastic Biochemical Processes: a Comparison of Moment Equation and Finite State Projection. In: The 10th International Workshop on Computational Systems Biology. (2013)

[69] Kermack, W.O., McKendrick, A.G.: A contribution to the mathematical theory of epidemics. In: Proceedings of the Royal Society of London, Series A. Volume 115., The Royal Society (1927) 700–721

[70] Komorowski, M., Finkenstädt, B., Harper, C., Rand, D.: Bayesian inference of biochemical kinetic parameters using the linear noise approximation. BMC bioinformatics **10** (2009)

[71] Koplan, J.P., Preblud, S.R.: A benefit-cost analysis of mumps vaccine. American Journal of Diseases of Children **136**(4) (1982) 362–364

[72] Kozine, I.O., Utkin, L.V.: Interval-Valued Finite Markov Chains. Reliable Computing **8**(2) (2002) 97–113

[73] Krishnarajah, I., Cook, A., Marion, G., Gibson, G.: Novel moment closure approximations in stochastic epidemics. Bulletin of Mathematical Biology **67**(4) (2005) 855–873

[74] Kurtz, T.G.: Solutions of ordinary differential equations as limits of pure jump Markov processes. Journal of Applied Probability **7**(1) (1970) 49–58

[75] Larsen, K.G., Skou, A.: Bisimulation through probabilistic testing. Information and Computation **94**(1) (1991) 1–28

[76] Li, Y., Liu, W., Turrini, A., Hahn, E.M., Zhang, L.: An Efficient Synthesis Algorithm for Parametric Markov Chains Against Linear Time Properties. In: SETTA. Lecture Notes in Computer Science (2016) To appear.

[77] Liu, J.S.: Monte Carlo strategies in scientific computing. Springer Science & Business Media (2008)

[78] Lloyd, J.R., Duvenaud, D., Grosse, R., Tenenbaum, J.B., Ghahramani, Z.: Automatic construction and Natural-Language description of nonparametric regression models. In: Association for the Advancement of Artificial Intelligence. (2014)

[79] Loinger, A., Biham, O.: Stochastic simulations of the repressilator circuit. Physical Review E **76** (Nov 2007) 051917

[80] Lyne, A.M., Girolami, M., Atchad, Y., Strathmann, H., Simpson, D.: On Russian Roulette Estimates for Bayesian Inference with Doubly-Intractable Likelihoods. Statistical Science **30**(4) (11 2015) 443–467

[81] Marco, D., Cairns, D., Shankland, C.: Optimisation of process algebra models using evolutionary computation. In: 2011 IEEE Congress on Evolutionary Computation (CEC). (2011) 1296–1301

[82] Marco, D., Scott, E., Cairns, D., Graham, A., Allen, J., Mahajan, S., Shankland, C.: Investigating Co-infection Dynamics through Evolution of Bio-PEPA Model Parameters: A Combined Process Algebra and Evolutionary Computing Approach. In Gilbert, D., Heiner, M., eds.: Computational Methods in Systems Biology. Volume 7605 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 227–246

[83] Marco, D., Shankland, C., Cairns, D.: Evolving Bio-PEPA process algebra models using genetic programming. In: Proceedings of the fourteenth international conference on Genetic and evolutionary computation. GECCO '12, New York, NY, USA (2012) 177–184

[84] Marin, J.M., Pudlo, P., Robert, C.P., Ryder, R.J.: Approximate Bayesian computational methods. Statistics and Computing **22**(6) (2012) 1167–1180

[85] Mateescu, M., Wolf, V., Didier, F., Henzinger, T.A.: Fast adaptive uniformisation of the chemical master equation. IET Systems Biology **4**(6) (November 2010) 441–452

[86] McGuire, P.: Getting Started with Pyparsing. First edn. O'Reilly (2007)

[87] McLeish, D.: A general method for debiasing a Monte Carlo estimator. Monte Carlo Methods and Applications (2011)

[88] McQuarrie, D.A.: Stochastic Approach to Chemical Kinetics. Journal of Applied Probability **4**(3) (1967) 413–478

[89] Milner, P., Gillespie, C.S., Wilkinson, D.J.: Moment closure based parameter inference of stochastic kinetic models. Statistics and Computing **23**(2) (2012) 287–295

[90] Milner, R.: Communication and Concurrency. Volume 84. Prentice Hall (1989)

[91] MIT Probabilistic Computing Project: Venture `http://probcomp.csail.mit.edu/venture/`.

[92] Moler, C., Loan, C.V.: Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later. SIAM Review **45**(1) (2003) 3–49

[93] Müller, A., Stoyan, D.: Comparison methods for stochastic models and risks. Volume 389. Wiley (2002)

[94] Munsky, B., Khammash, M.: The finite state projection algorithm for the solution of the chemical master equation. The Journal of Chemical Physics **124**(4) (2006)

[95] Munsky, B., Khammash, M.: A multiple time interval finite state projection algorithm for the solution to the chemical master equation. Journal of Computational Physics **226**(1) (2007) 818–835

[96] Murray, I., Graham, M.M.: Pseudo-Marginal Slice Sampling. Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, Journal of Machine Learning Research Workshop & Conference Proceedings **51** (2016) 911–919

[97] National Health Service: Mumps `http://www.nhs.uk/Conditions/Mumps/Pages/Introduction.aspx` [accessed July 22 2016].

[98] Nori, A.V., Hur, C.K., Rajamani, S.K., Samuel, S.: R2: An Efficient MCMC Sampler for Probabilistic Programs. In: AAAI Conference on Artificial Intelligence (AAAI), AAAI (July 2014)

[99] Oaken, D.R.: Optimisation of Definition Structures & Parameter Values in Process Algebra Models Using Evolutionary Computation. PhD thesis, University of Stirling (2014)

[100] Opper, M., Sanguinetti, G.: Variational inference for Markov jump processes. In Platt, J., Koller, D., Singer, Y., Roweis, S., eds.: Advances in Neural Information Processing Systems 20. MIT Press, Cambridge, MA (2008) 1105–1112

[101] Owen, J., Wilkinson, D.J., Gillespie, C.S.: Scalable inference for Markov processes with intractable likelihoods. Statistics and Computing **25**(1) (2014) 145–156

[102] Papaspiliopoulos, O.: Monte Carlo probabilistic inference for diffusion processes: a methodological framework. Bayesian time series models (2011) 82–99

[103] Pedersen, A.R.: A New Approach to Maximum Likelihood Estimation for Stochastic Differential Equations Based on Discrete Observations. Scandinavian Journal of Statistics **22**(1) (1995) 55–71

[104] Petri, C.A.: Communication with automata. PhD thesis, Universität Hamburg (1966)

[105] Pfeffer, A.: The Design and Implementation of IBAL: A General-Purpose Probabilistic Language. In Getoor, L., Taskar, B., eds.: Introduction to Statistical Relational Learning. The MIT Press (2007)

[106] Pfeffer, A.: CTPPL: A Continuous Time Probabilistic Programming Language. In: IJCAI. (2009) 1943–1950

[107] Pourranjbar, A., Hillston, J., Bortolussi, L.: Don't Just Go with the Flow: Cautionary Tales of Fluid Flow Approximation. In Tribastone, M., Gilmore, S., eds.: Computer Performance Engineering: EPEW 2012 and UKPEW 2012, Revised Selected Papers, Springer Berlin Heidelberg (2013) 156–171

[108] Priami, C., Quaglia, P.: Operational patterns in Beta-binders. Transactions on Computational Systems Biology **1** (2005) 50–65

[109] Rao, V., Teh, Y.W.: Fast MCMC sampling for Markov jump processes and continuous time Bayesian networks. In: Proceedings of the International Conference on Uncertainty in Artificial Intelligence. (2011)

[110] Rao, V., Teh, Y.W.: Fast MCMC sampling for Markov jump processes and extensions. Journal of Machine Learning Research **14** (2013) 3207–3232 arXiv:1208.4818.

[111] Rhee, C.H., Glynn, P.W.: Unbiased estimation with square root convergence for SDE models. Operations Research **63**(5) (2015) 1026–1043

[112] Ross, B.J., Imada, J.: Evolving stochastic processes using feature tests and genetic programming. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, ACM (2009) 1059–1066

[113] Ruttor, A., Opper, M.: Efficient Statistical Inference for Stochastic Reaction Processes. Physical Review Letters **103**(23) (2009)

[114] Sanguinetti, G., Ruttor, A., Opper, M., Archambeau, C.: Switching regulatory models of cellular stress response. Bioinformatics **25**(10) (2009) 1280–1286

[115] Schnoerr, D., Sanguinetti, G., Grima, R.: Validity conditions for moment closure approximations in stochastic chemical kinetics. The Journal of Chemical Physics **141**(8) (2014)

[116] Schnoerr, D., Sanguinetti, G., Grima, R.: Comparison of different moment-closure approximations for stochastic chemical kinetics. The Journal of Chemical Physics **143**(18) (2015)

[117] Sciacca, E., Spinella, S., Calcagno, C., Damiani, F., Coppo, M.: Parameter Identification and Assessment of Nutrient Transporters in AM Symbiosis through Stochastic Simulations. Electronic Notes in Theoretical Computer Science **293**(0) (2013) 83–96 Proceedings of CS2Bio'12.

[118] Sen, K., Viswanathan, M., Agha, G.: Model-Checking Markov Chains in the Presence of Uncertainties. In Hermanns, H., Palsberg, J., eds.: Tools and Algorithms for the Construction and Analysis of Systems. Volume 3920 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2006) 394–410

[119] Sherlock, C., Thiery, A.H., Roberts, G.O., Rosenthal, J.S.: On the efficiency of pseudo-marginal random walk Metropolis algorithms. Annals of Statistics **43**(1) (02 2015) 238–275

[120] Spiegelhalter, D.J., Thomas, A., Best, N.: Computation on Bayesian graphical models. Bayesian statistics **5**(5) (1996) 407–425

[121] Stathopoulos, V., Girolami, M.A.: Markov chain Monte Carlo inference for Markov jump processes via the linear noise approximation. Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences **371**(1984) (2012)

[122] Takla, A., Wichmann, O., Rieck, T., Matysiak-Klose, D.: Measles incidence and reporting trends in Germany, 2007–2011. Bulletin of the World Health Organization **92**(10) (2014) 742–749

[123] Tian, T., Burrage, K.: Stochastic models for regulatory networks of the genetic toggle switch. Proceedings of the National Academy of Sciences **103**(22) (2006) 8372–8377

[124] Tierney, L., Kadane, J.B.: Accurate Approximations for Posterior Moments and Marginal Densities. Journal of the American Statistical Association **81**(393) (1986) 82–86

[125] Toni, T., Welch, D., Strelkowa, N., Ipsen, A., Stumpf, M.P.: Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. Journal of The Royal Society Interface **6**(31) (2009) 187–202

[126] Tracol, M., Desharnais, J., Zhioua, A.: Computing distances between probabilistic automata. In: Proceedings Ninth Workshop on Quantitative Aspects of Programming Languages (QAPL). (2011)

[127] Tribastone, M., Gilmore, S., Hillston, J.: Scalable Differential Analysis of Process Algebra Models. IEEE Transactions on Software Engineering **38**(1) (Jan 2012) 205–219

[128] van der Walt, S., Colbert, S.C., Varoquaux, G.: The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science Engineering **13**(2) (March 2011) 22–30

[129] van Moorsel, A.P.A., Sanders, W.H.: Adaptive uniformization. Stochastic Models **10**(3) (1994) 619–647

[130] Wilkinson, D.J.: Stochastic Modelling for Systems Biology. CRC Press (2011)

[131] Williams, C.D., Hillston, J.: Automated Capacity Planning for PEPA Models. In Horvth, A., Wolter, K., eds.: Computer Performance Engineering. Volume 8721 of Lecture Notes in Computer Science. Springer International Publishing (2014) 209–223

[132] Wood, F., van de Meent, J.W., Mansinghka, V.: A New Approach to Probabilistic Programming Inference. In: Proceedings of the 17th International conference on Artificial Intelligence and Statistics. (2014) 1024–1032

[133] Zechner, C., Unger, M., Pelet, S., Peter, M., Koeppl, H.: Scalable inference of heterogeneous reaction kinetics from pooled single-cell recordings. Nature Methods **11**(2) (2014) 197–202